# 《LINUX 就该这么学》

此书版本 V1.01　（每隔 30 天更新一次）

最新版下载地址：HTTP://WWW.LINUXPROBE.COM/book

尊敬的学员：

感谢您报名参加 LinuxProbe.com 在线培训，现在班型类型包括有晚班、周末班及脱产班，您可以根据自身情况来选择最合适的班型，我们的培训采用在线培训的方式，如果您因故不能参加某天的培训，请不要着急，我们已经为您录制好了当期的培训视频，直接联系讲师或在内部资料区下载即可，我们竭诚为您提供最高质量的 Linux 在线培训课程，珍惜您对我们的信任，本教材内含大量的空白页，并附赠一只黑笔，一只蓝笔及一只红笔，方便您随着上课写好笔记，另外如果学习中有任何的建议都请直接告诉我们，我们超爱您的建议！

刘遄

您的终身职业顾问及技术导师

# 第 0 章 让我们谈谈学习方法和红帽系统。

**章节简述：**

Hello World!全书的开篇讲述作者学习红帽 Linux 系统的经验以及写书过程的感悟，分析学习 Linux 的目地与意义。开源精神是种让每个从事 Linux 行业的技术人从骨子里自豪的情怀，开源产品的兴盛受益于开源社区强健的根基。优秀的 Linux 运维师能够让用户真切体会到 Linux 系统带来的高可用、高性能与安全稳定。

## 0.1 本书作者简介

本书作者刘遄从事于 linux 运维技术行业，较早时因兴趣的驱使接触到了 Linux 系统并开始学习，已在 2012 年考下红帽工程师 RHCE_6，今年又分别考下 RHCE_7 版本与红帽架构师认证 RHCA，深知水平有限且技术一般，若没有得益于良师益友的无私帮助，肯定不能如此顺利的完成 Linux 学业，同样作为一名普通的技术人，我亲身经历过半夜还在培训班的心酸，体验过拥堵 6 小时车程的无奈，所以为了能够帮助读者们快速入门 Linux 系统，此刻我正怀揣着一颗忐忑的心，竭尽全身心的斗志将书编写的更好。

本书于 2015 年的春节前夕起笔，预计年末截稿（初版）——为了保证每篇文章的质量所以很可能会写不完，才与诚合，然后事可成，恃才而败。我将付出不亚于任何人的努力，与可爱的读者们一起编写、完善这本书籍，带领大家从"0"基础开始学习 linux 系统，配以大量 Linux 相关实验逐步掌握运维之道，本书内含配套教学图片与视频，达到增强学员兴趣与加深记忆的作用，当然都是免费的，主动抛弃不实用的部分，将重点反复实践，所以尤其适合希望尽快掌握 Linux 系统的人群。

## 0.2 学习是件苦差



我无意回避这个问题——学习本是件痛苦的事情，如果学习 Linux 真的很简单，那么必是骗子说的谎话，起码这将不能给你带来高薪，打开电脑后的沉思，是该聊会天那~还是追个美剧那~还是打盘 LOL 那~还是看看那该死的刘遄写的那本可怕 Linux 教材时，请不要忘记自己最初的梦想，十年后你会感谢此时正在努力学习的自己。我身为作者的使命就是一定要对得起您花费的时间、精力、金钱，让您每学完一个章节都是一次进步，**读完稻盛和夫先生的活法后发现"我们也可以从学习中获得快乐"。**

"工作马马虎虎，只想在兴趣和游戏中寻觅快活，充其量只能获得一时的快感，绝不能尝到从心底涌出的惊喜和快乐，但来自工作的喜悦并不像糖果那样——放进嘴里就甜味十足，而是需要从苦劳与艰辛中渗出，因此当我们聚精会神，孜孜不倦，克服艰辛后的成就感，世上没有哪种喜悦可以类比。"

"更何况人类生活中工作占据了较大的比重，如果不能从劳动中、工作中获得充实感，那么即使从别的地方找到快乐，最终我们仍然会感到空虚和缺憾。"

## 0.3 开源共享精神

坦白来讲，每个从事 Linux 行业的技术人都从骨子里有一种独特的情怀，听到开源产品的兴起就会由衷的自豪，开源企业不单纯为了利益，而是互相扶持，让开源软件越来越完善，根基越来越强大，开源社区越来越有人气，开源软件简单来说就是可以不受限制的使用某个软件并且随意修改，甚至修改成自己的产品再发布出去。所以开源软件一般会将软件程序与源代码一起提供给用户，最热门的六种开源许可证包括：




开源软件的特性："**使用自由**"，"**修改自由**"，"**重新发布自由**"，"**创建衍生品自由**"。

### 0.4 为什么要学 Linux?

Linux 操作系统最初是在1991年10月份由芬兰赫尔辛基大学的在校生Linus Torvalds所发布，最初被发布的 LINUX 0.02 版本因其高质量的代码与开放源代码，迅速引起了一大批黑客的加入，而今虽然有数百计的 Linux 发布版，但都依然统一使用 Linus Torvalds 开发/维护的**系统内核**，Linux 是具有类似 Unix 的程序界面与操作方法且继承了其稳定性(通常运行几年都不会宕机)。




大多数读者开始了解计算机和网络都是从"Windows™"开始的吧，肯定已经习惯了盖茨系统而且觉得足以应付日常工作啦。虽然盖茨系统确实很优秀但同时也是用户对**安全性、高可用与高性能**的大大牺牲，因为你一定见过右面的图片。

所以读者是否考虑过为何需要长期稳定运行的网站服务器、处理大数据的集群系统或者需要协同工作的环境大多采用 Linux 系统呢？




**LinuxPKwindows**

稳定且有效率

免费或少许费用

漏洞少且快速修补

多任务多用户

更加安全的用户及文件权限策略

适合小内核程序的嵌入系统

相对不耗资源

**Linux 的优势读者可先作了解暂不需深究，学习中再慢慢感受。**

## 0.5 热门的开源系统

**红帽企业系统（RedHatEnterpriseLinux,RHEL.）**
全球最大的开源技术厂商，全世界内使用最广泛的 Linux 发布套件，
提供性能与稳定性极强的 Linux 套件系统并拥有完善的全球技术支持。

**社区企业操作系统（Centos）**
最初是将红帽企业系统“重新编译/发布”给用户免费使用而广泛使用，
当前已正式加入红帽公司并继续保持免费（随 RHEL 更新而更新）。

**红帽用户桌面版（Fedora [Linux]）**
最初由红帽公司发起的桌面版系统套件（目前已经不限于桌面版），
用户可免费体验到最新的技术或工具，而功能成熟后加入到 RHEL 中。

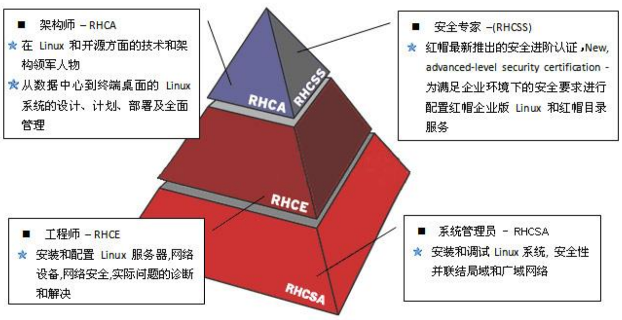
**国际化组织的开源操作系统（Debian）**
提供超过 37500 种不同的自由软件且拥有很高的认可度，对于各类内核架构支持性良好，稳定性、安全性强更有免费的技术支持。

**基于 Debian 的桌面版（Ubuntu）**
Ubuntu 是一款基于 Debian 派生的产品，对新款硬件具有极强的兼容能力。
普遍认为 Ubuntu 与 Fedora 都是极其出色的 LINUX 桌面系统。

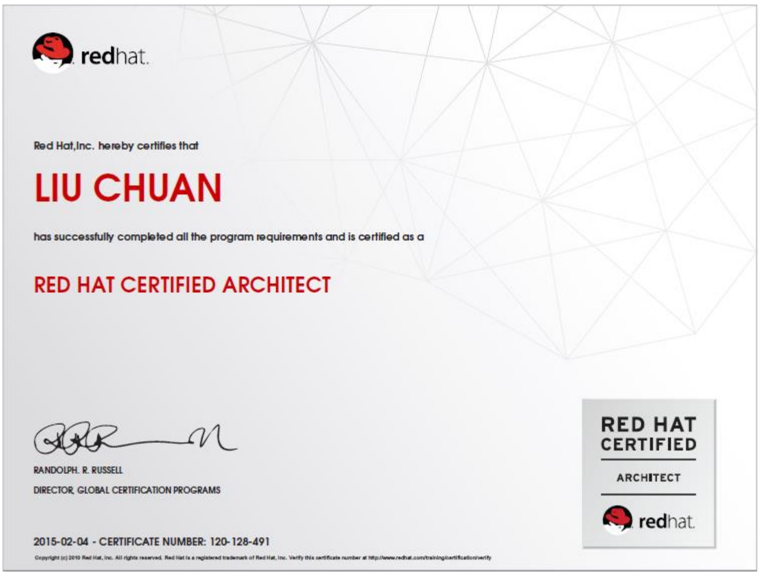
## 0.6 认识红帽认证

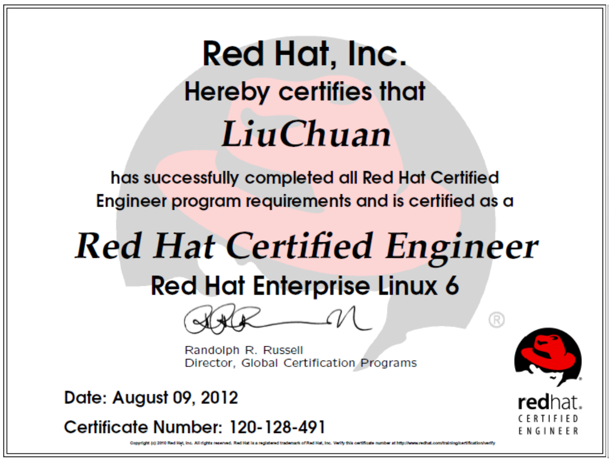
Linux 系统有上百个不同的组织、公司、机构研发并发布出不同的版本，其中红帽公司作为一家成熟的操作系统厂商提供可靠的 Linux 系统和完善的求援服务，红帽企业 linux 系统（RedHat Enterprise Linux,RHEL）的市场占有量极大，认可度也非常高。

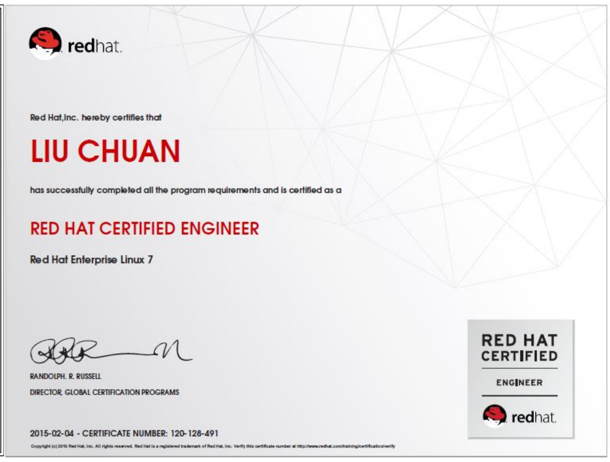
**红帽公司推出了阶梯式的认证体系也确实能够帮助读者检查自己的能力：**



同于大家的了解，本书《Linux 就该这么学》就是由一批中国的红帽架构师所编写。




本章结束，您可以在此写下笔记：

第 1 章 部署虚拟环境安装 linux 系统。

**章节简述:**

本章节带领读者从 0 基础了解虚拟机与红帽系统，完整的演示了在 VM 与 KVM 中安装红帽 RHEL7 系统的方法。特别增加了超级实用的 Linux 系统找回 root 密码、虚拟机功能增强包、VNC 远程控制服务等相关的技术知识点。简单了解守护进程即可，对了！在安装 RPM 软件包或配置 YUM 软件仓库时请格外注意参数细节哦~

## 1.1 准备您的工具

所谓工欲善其事必先利其器，在本书第一章需要读者们搭建出为课后练习实验所使用的红帽 RHEL7 系统环境，读者不需要为了课程实验而单独购买一台新电脑，下面的小节中会教给您如何通过“虚拟机”来模拟出“仿真系统”，虚拟机是能够让用户在一台真机上模拟出多台操作系统的软件，一般来讲当前主流的硬件配置都是没问题的。

**强烈建议读者采用与本书一致的虚拟机软件与 RHEL7 镜像系统，否则可能会导致实验失败!!**

**软件资源请在这里下载:**http://www.linuxprobe.com/tools/

VmwareWorkStation 11.0——**虚拟机软件（必需）:**

功能强大的桌面虚拟计算机软件，能够让用户在单一主机同时运行多个不同的操作系统。

同时支持实时快照，虚拟网络，拖拽文件以及 PXE 等强悍功能。

RedHatEnterpriseLinux [RHEL]7.0——**红帽操作系统（必需）:**

由开源软件及全球服务性系统开发商红帽公司出品，最稳定出色的 Linux 操作系统。

说来真的很郁闷、其实我在高中时就有学习 Linux 系统的冲动，那时上网还不便捷，所以安装系统都需要去买光盘，而那时的 linux 系统至少需要 6 张光盘（CD-Rom 容量为 700M)，尝试安装了几次却一直报错，搞不懂只能放弃了，今年春节收拾屋子翻出了这些光盘，再次安装时找到了错误的原因，原来是第五张光盘被“刮花”了，导致依赖的软件包无法安装，真的是很无语、很郁闷，原本可以早几年就开始学 Linux 系统了，所以这里提示读者：“**准备齐工具后一定要校验完整性**”。

Hash1.0.4——**文件校验工具（推荐）:**

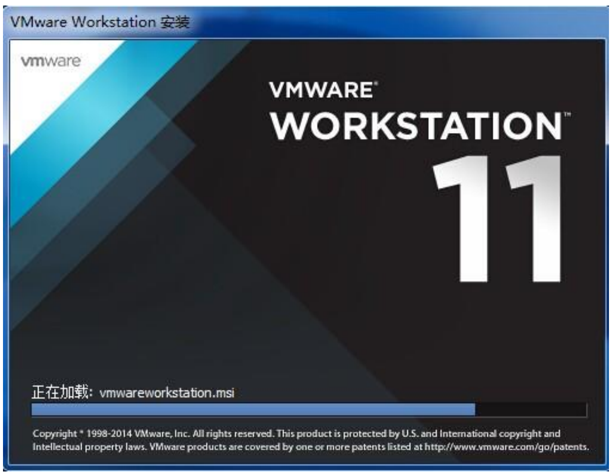
经典实用的功能且便捷的支持文件拖拽查询，确保文件的完整与安全性，只需选中文件或直接拖拽进去，确保你在 Hash 界面上看得到 MD5 与 SHA1 值与软件资源库提供的一致再使用。

## 1.2 安装配置虚拟机

Vmware WorkStation 是一款桌面计算机虚拟软件，能够让用户**在单一主机上同时运行多个不同的操作系统**。每个虚拟操作系统的硬盘分区、数据配置都是独立的，同时又可以将多台虚拟机构建为一个局域网。更何况 Linux 系统要求的系统资源很低，所以读者们真的没有必要再买一台电脑，课程实验完全可以用虚拟机搞定，而且 VM 还支持实时快照、虚拟网络、拖拽文件以及 PXE 等方便实用功能。

**执行虚拟机软件安装向导**

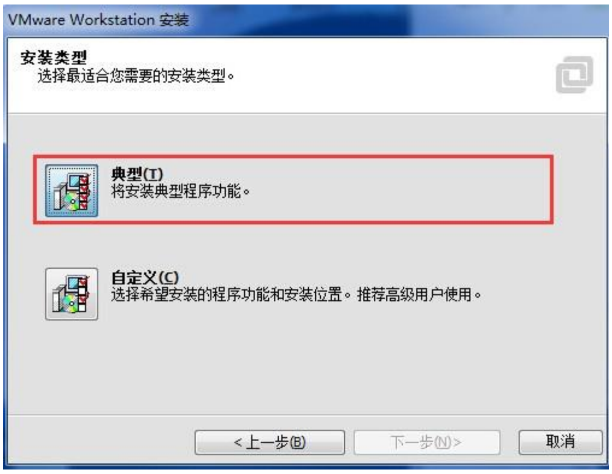
第 1 步:运行虚拟机软件。




第 2 步:接受软件的许可。



第 3 步:选择典型安装。




第 4 步:选择安装到的目录。



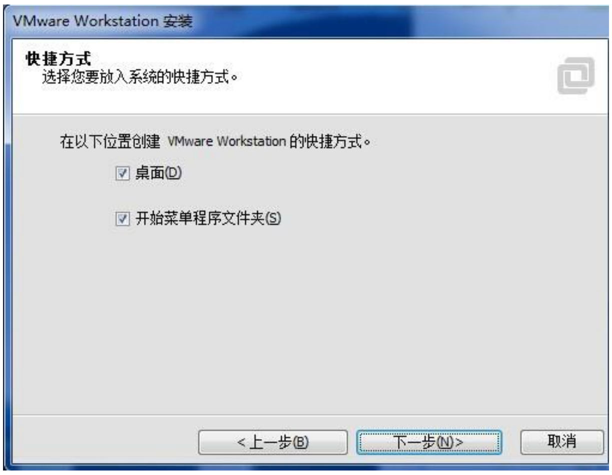

第 5 步:自动检查新版。




第 6 步:帮助改进虚拟机软件。




第 7 步:在桌面上创建图标（快捷方式）。



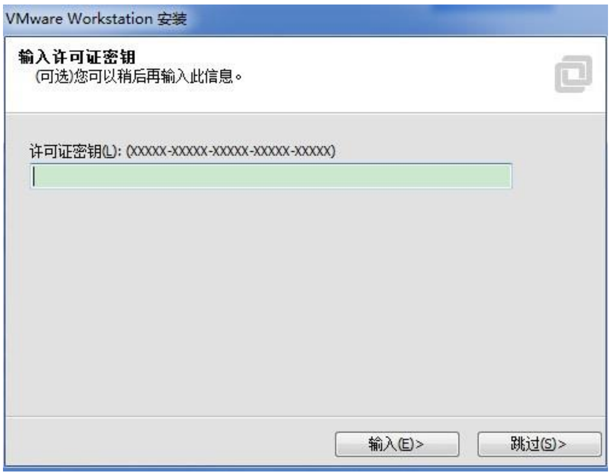

第 8 步:不错,一切都准备就绪了。



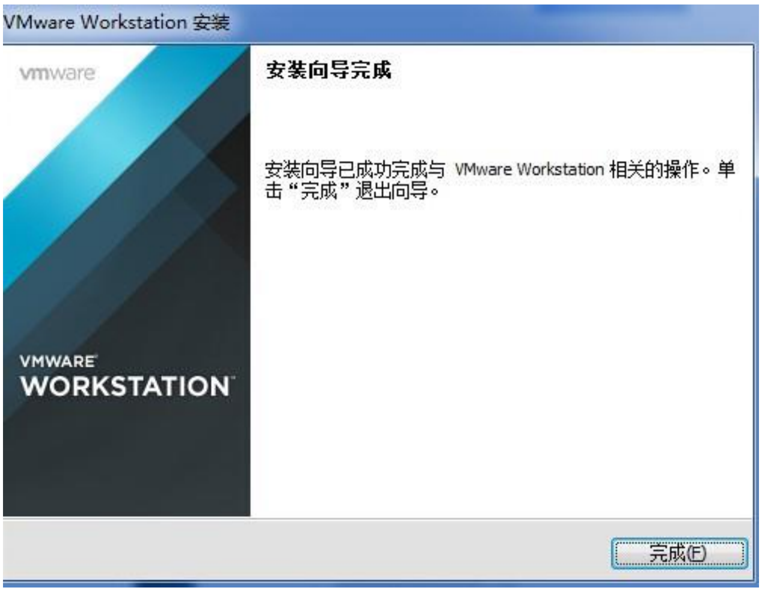
第 9 步:正常安装中。




第 10 步:请填写密钥或直接跳过。

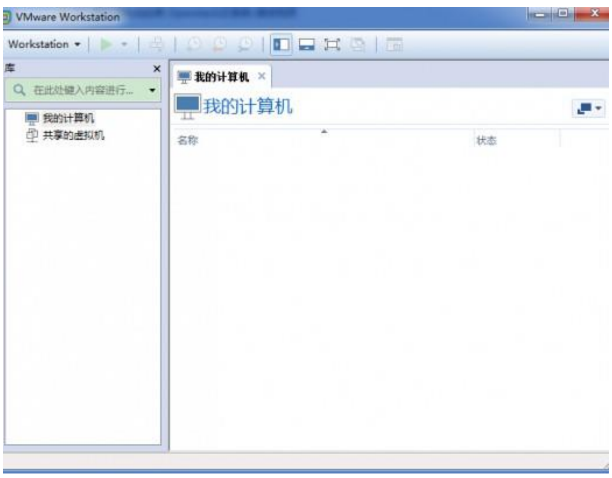



第 11 步:安装顺利完成,Good Job!

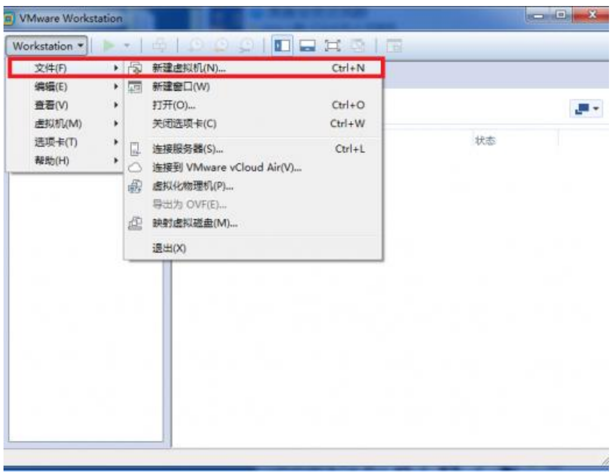



**模拟出用于安装 RHEL7 红帽操作系统的硬件配置。**

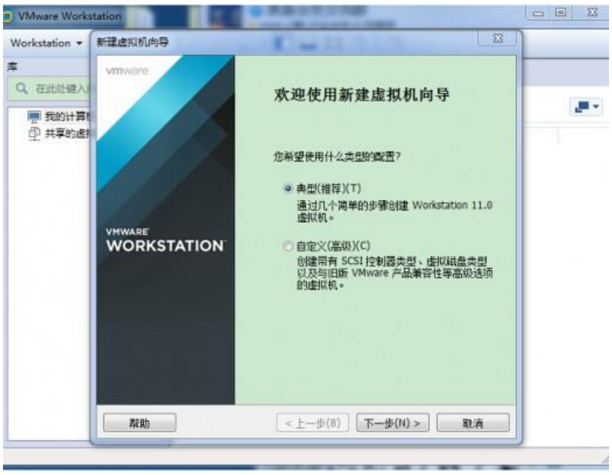
第 1 步：运行"Vmware WorkStation"，看到主页面。



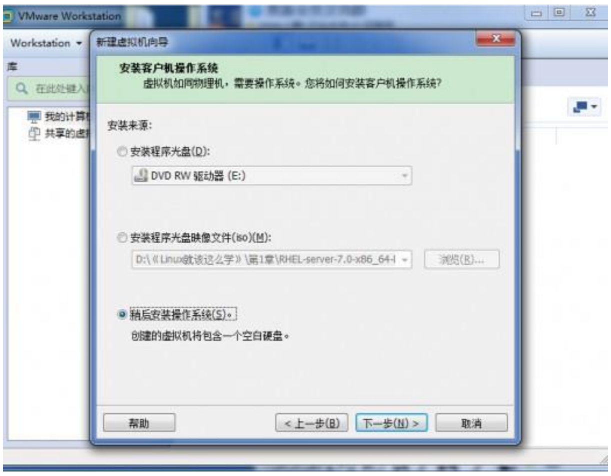

第 2 步：创建新的虚拟机。



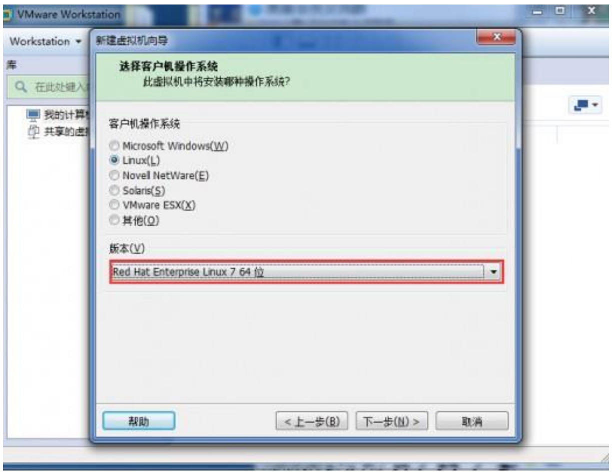
第 3 步：新建虚拟机向导——典型（推荐）。



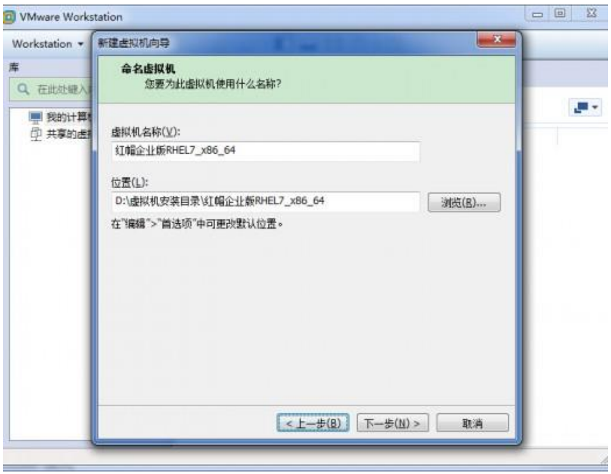

第 4 步：选择稍后安装操作系统。




第 5 步：定义版本为"Red Hat Enterprise Linux 7 64 位"。




第 6 步：设置虚拟机名称与安装路径。



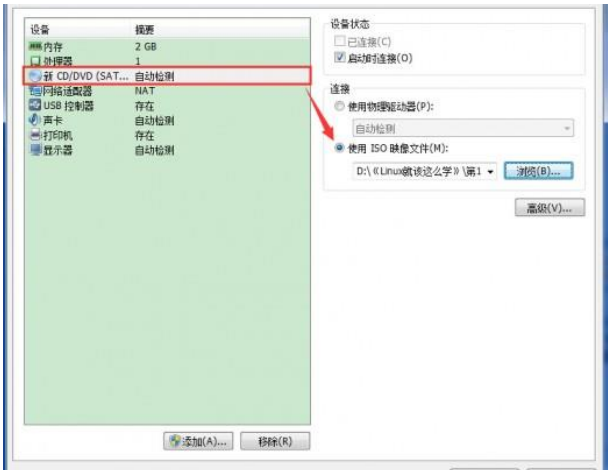

第 7 步：设置磁盘为 20GB 即可（足够了）。




第 8 步：完成向导后请点击“自定义硬件”。



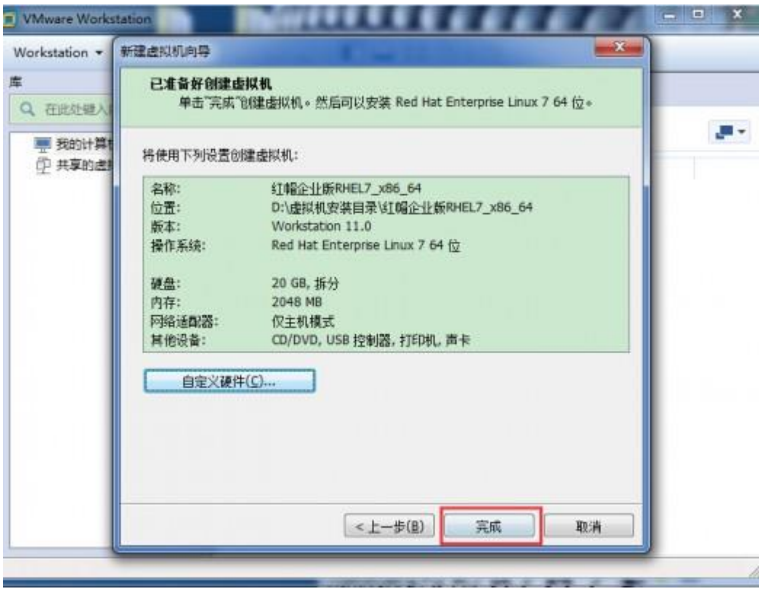
第 9 步：选择“设置光驱”，选择到 RHEL7 镜像。



第 10 步:选择“设置网络适配器”为“仅主机模式”。



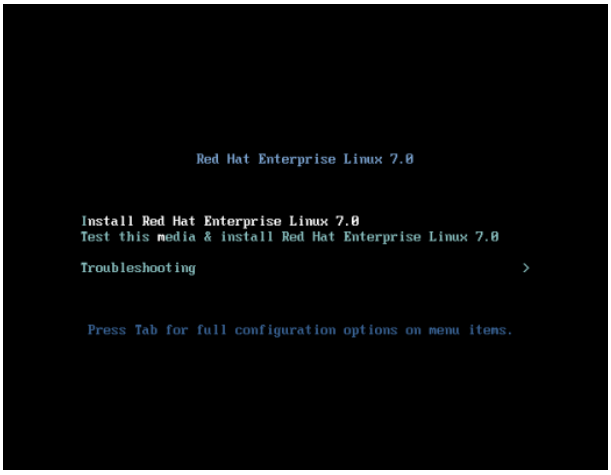
第 11 步：全部设置完成,请点击“完成”。




## 1.3 VM 安装 RHEL7 系统

通过上面小节已经将虚拟机配置完毕，现在正式安装红帽 RHEL7 系统啦。

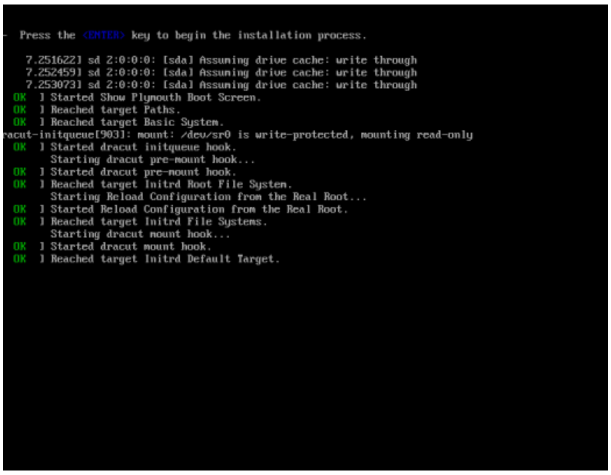
第 1 步:启动 RHEL7 的主机电源。



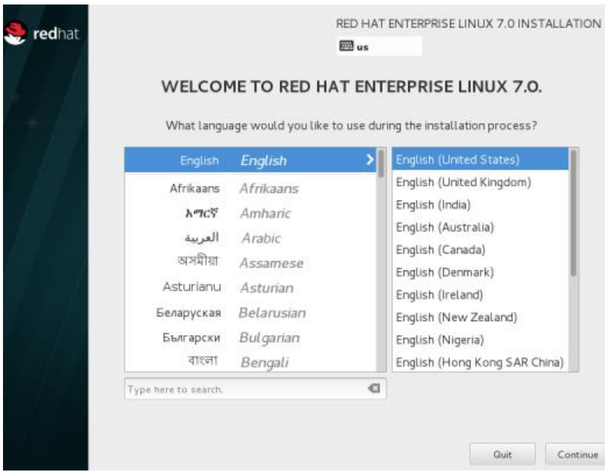

第 2 步:敲击回车。



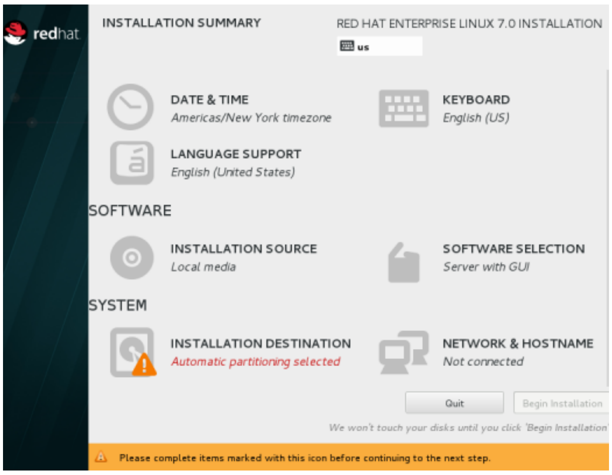
第 3 步:等待即可。



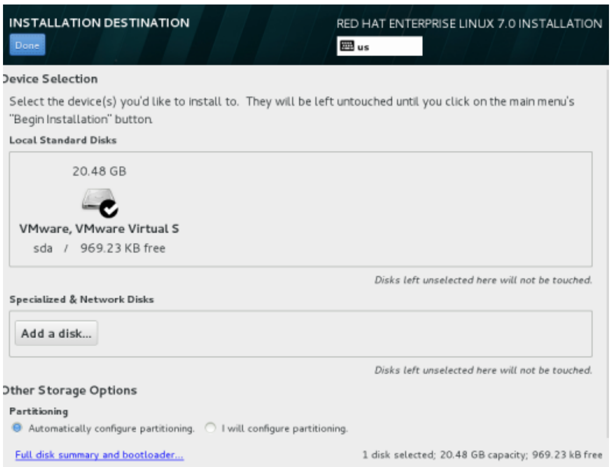

第 4 步:选择安装系统时的语言。



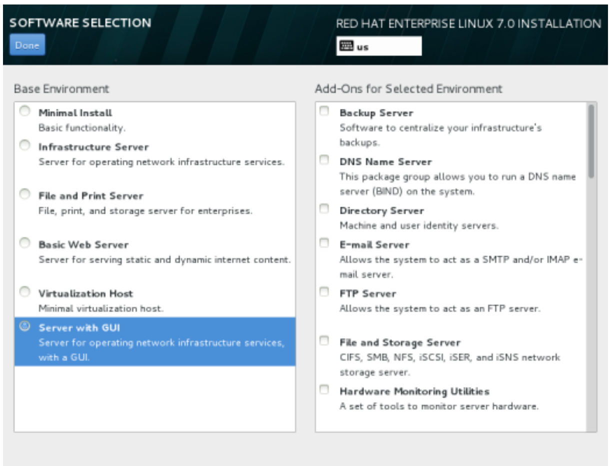

第 5 步:配置信息界面,敲击“Installation Destination”。



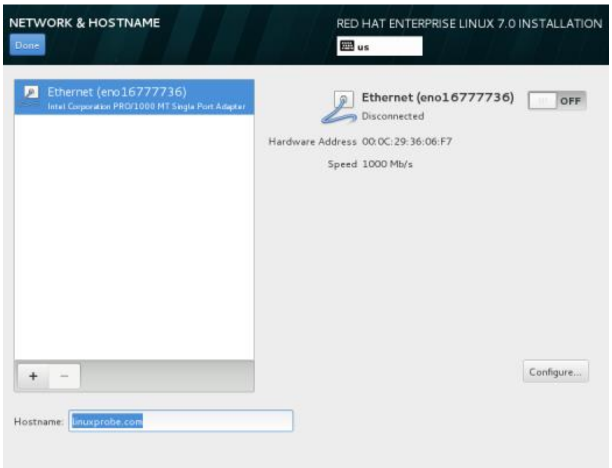

第 6 步:进入后选择硬盘并点击左上角“Done”。



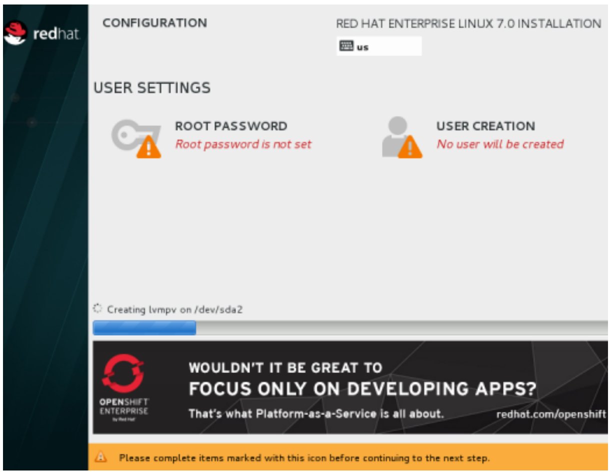

第 7 步:返回主页面后再点击“Software Selection”后选择"Server With GUI"。



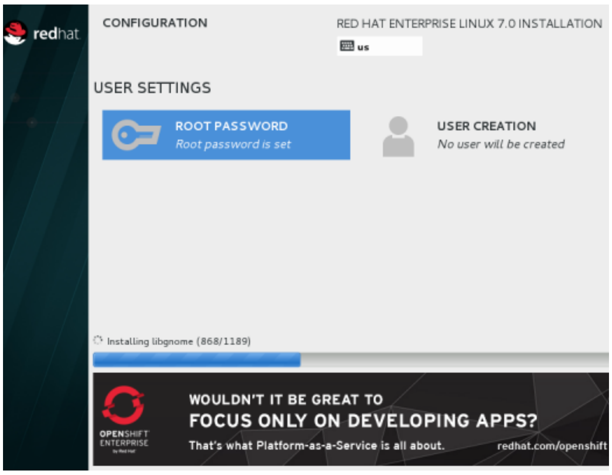
第 8 步:返回主页面后再点击"Network & Hostname"后设置主机名"linuxprobe.com"。



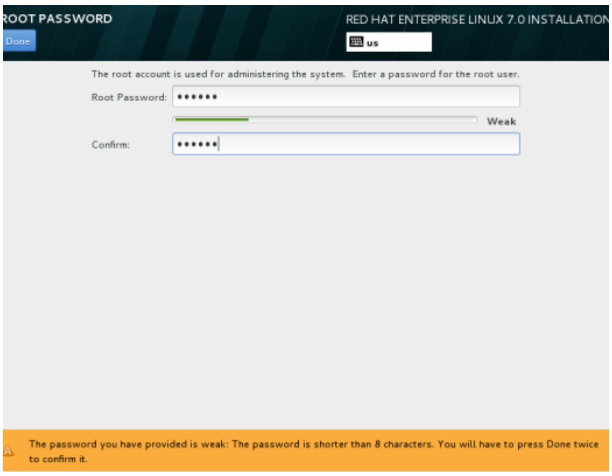

第 9 步:一切就绪后返回主页面并点击“Begin Installation”。



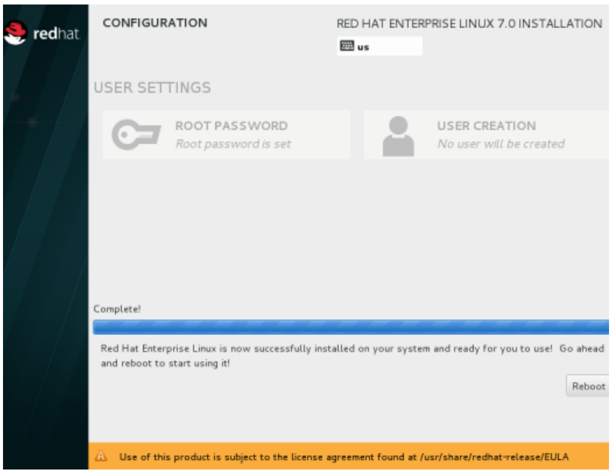

第 10 步:点击“Root Password”。




第 11 步:设置 Root 用户的密码（简单密码请双击 Done）

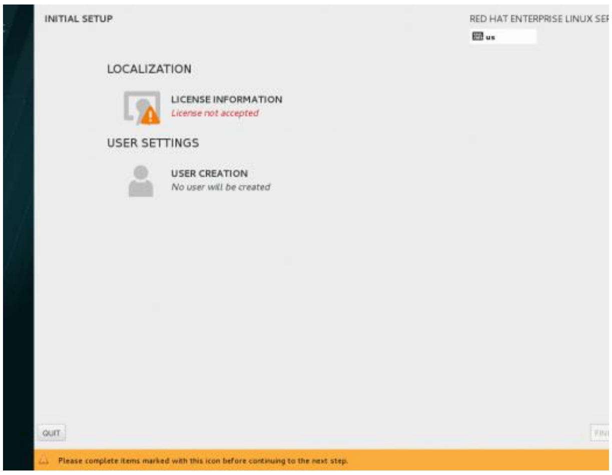


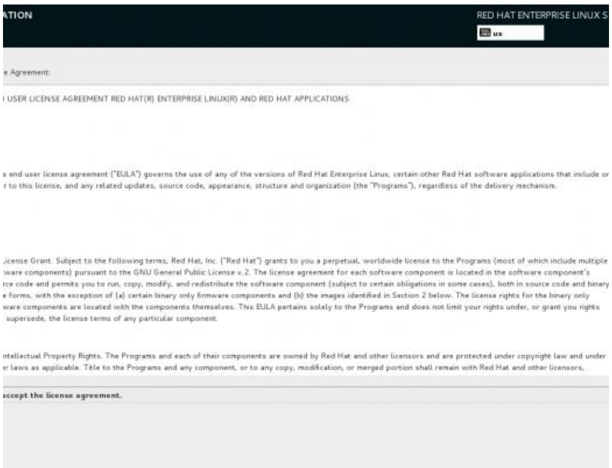

第 12 步:等待安装完成后点击“Reboot”。



第 13 步:重启后选择“License Instormation”。

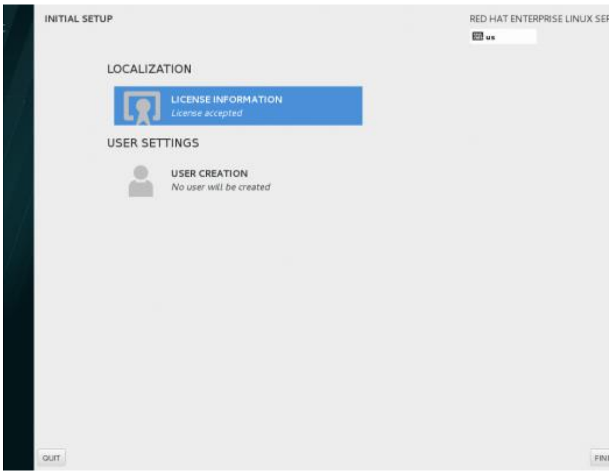
第 14 步:选中“I accept the license agreement”后敲击“Done”。

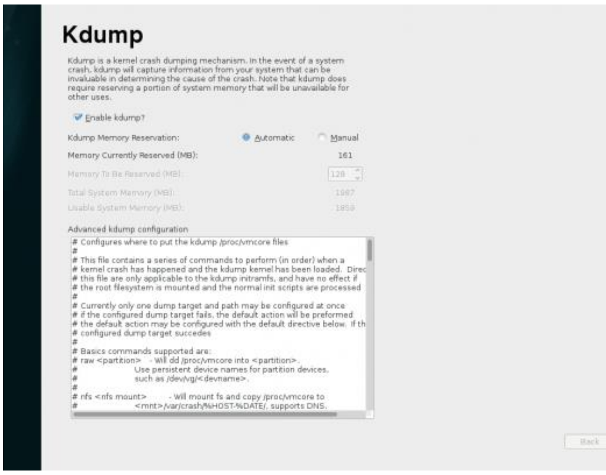


第 15 步:敲击“Finish Configuration”。

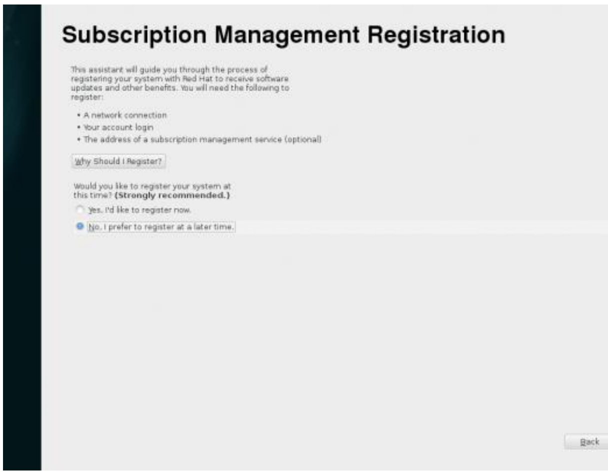
第 16 步:为本书后章讲到的“Kdump”建议开启(默认)。



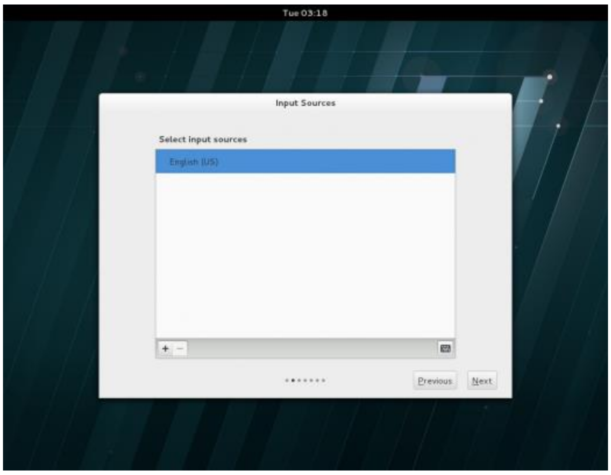

第 17 步:选择“No.I prefer to register at later time.”。

第 18 步:选择系统语言,(本书例题用英文版完成)。





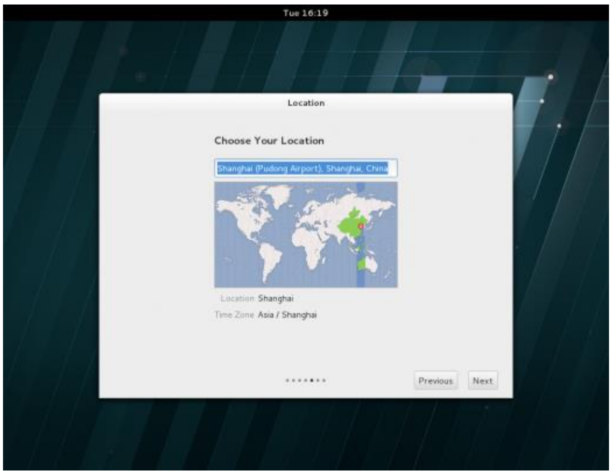

第 19 步:选择输入资源，默认即可。



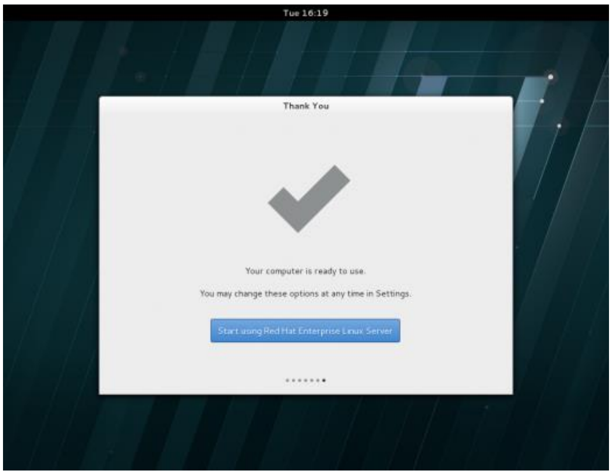

第 20 步:创建一个本地用户（权限比 Root 小，但更加安全）。



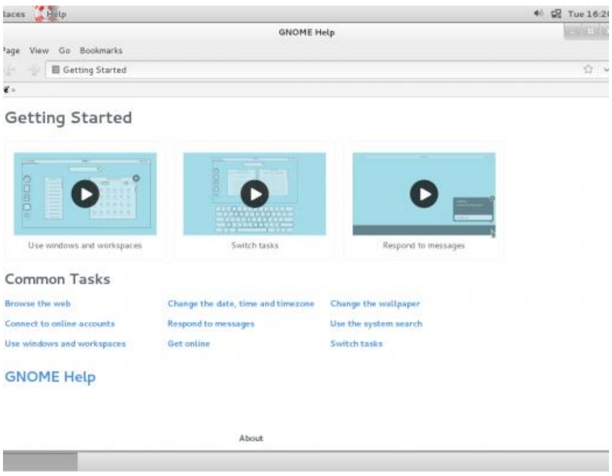

第 21 步:设置系统时间（上海.中国）。




第 22 步:选择“Start using Red Hat Enterprise Linux Server”。




第 23 步:恭喜，您已经顺利的安装了红帽 RHEL7 操作系统。




**或使用 KVM 安装系统**

VmwareWorkstation 可并不是唯一能够实现虚拟化的软件，如果您之前学习过 Linux 系统或有一定的命令行基础，并且想试试在 linux 系统中安装其他系统，现在就教给您操作方法，新手读者请先跳过这个小节吧~

KVM(Kernel Virtual Module)能够提供像 Vmware 一样的全虚拟化功能——**让虚拟机用起来跟真实物理机一摸一样**。

**安装 KVM 之前我们要检查真实物理机是否支持虚拟化功能:**

[root@linuxprobe ~]# grep vmx /proc/cpuinfo
flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts mmx fxsr sse sse2 ss syscall nx rdtscp lm constant_tsc arch_perfmon pebs bts nopl xtopology tsc_reliable nonstop_tsc aperfmperf pni pclmulqdq vmx ssse3 cx16 pcid sse4_1 sse4_2 x2apic popcnt aes xsave avx f16c rdrand hypervisor lahf_lm ida arat epb pln pts dtherm tpr_shadow vnmi ept vpid fsgsbase smep

如果执行该命令后没有输出 vmx 或 svm 值,但您的电脑是近几年买来的,那么很有可能只是在 BIOS 中默认关闭了,请去开启试试吧!

Inter 处理器的虚拟技术标志为 vmx。

AMD 处理器的虚拟技术标志为 svm。

**安装 KVM 以及相关的依赖软件包:**

[root@linuxprobe ~]# yum -y groupinstall "Virtualization Host"
[root@linuxprobe ~]# yum -y groupinstall "Virtualization Host"
Loaded plugins: langpacks, product-id, subscription-manager
………………省略部分安装过程………………
Complete!
[root@linuxprobe ~]# yum -y install virt-{install,viewer,manager}
Loaded plugins: langpacks, product-id, subscription-manager
………………省略部分安装过程………………
Complete!

**为了让 KVM 中虚拟机能够互相共享数据，还必需配置真实机的网络:**

让系统支持 ipv4 的转发功能:

[root@linuxprobe ~]# echo "net.ipv4.ip_forward = 1" > /etc/sysctl.d/99-ipforward.conf

让转发功能立即生效:

[root@linuxprobe ~]# sysctl -p /etc/sysctl.d/99-ipforward.conf
 net.ipv4.ip_forward = 1

**新手读者们请注意，前方高能:**

如果您还没有学习过 Linux 命令，Vim 编辑器与网卡配置方法的话请先跳过本小节!!

将网卡配置文件中的 IP 地址、子网掩码等信息注释后追加参数 BRIDGE=virbr0(设置网卡为桥接模式):

[root@linuxprobe ~]# vim /etc/sysconfig/network-scripts/ifcfg-eno16777736
 DEVICE="eno16777736"
 ONBOOT=yes
 #IPADDR="192.168.10.10"
 #NETMASK="255.255.255.0"
 #GATEWAY="192.168.10.1"
 HWADDR="网卡的 MAC 地址"
 #DNS1="192.168.10.1"
 BRIDGE=virbr0

创建用于桥接网卡的配置文件（与上面的配置文件很相似）:

[root@linuxprobe ~]# vim /etc/sysconfig/network-scripts/ifcfg-virbr0
 DEVICE="virbr0"
 TYPE=BRIDGE
 ONBOOT=yes
 BOOTPROTO=static

IPADDR="192.168.10.10"
NETMASK="255.255.255.0"
GATEWAY="192.168.10.1"
DNS1="192.168.10.1"

**当 KVM 安装完成并将网卡配置妥当后请重启(reboot)后再进行下面的检查操作:**

检查 kvm 模块是否被加载以及能否正常的使用 CPU 虚拟化功能:

[root@linuxprobe ~]# lsmod | grep kvm
kvm_intel 138567 0
kvm 441119 1 kvm_intel

检查桥接的网卡配置是否启用成功:

[root@linuxprobe ~]# ip show virbr0
3: virbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
link/ether 00:0c:29:9c:63:73 brd ff:ff:ff:ff:ff:ff
inet 192.168.10.10/24 brd 192.168.10.255 scope global virbr0
valid_lft forever preferred_lft forever
inet6 fe80::20c:29ff:fe9c:6373/64 scope link
valid_lft forever preferred_lft forever

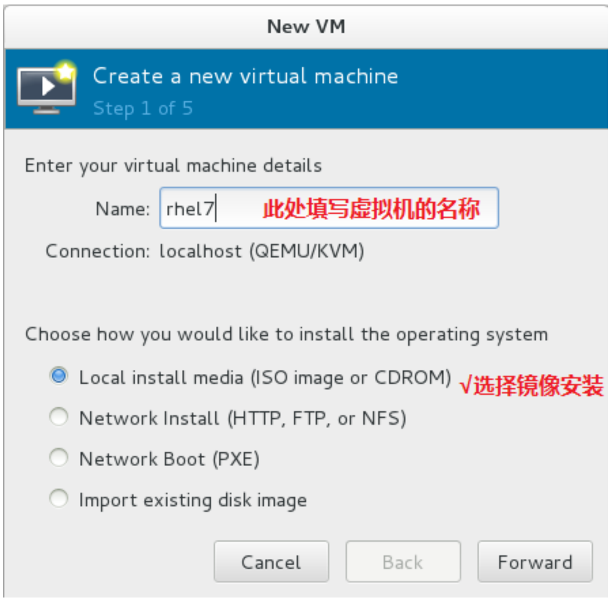
获取虚拟机列表（默认为空是正常的）：

[root@linuxprobe ~]# virsh -c qemu:///system list
Id Name State
----------------------------------------------------

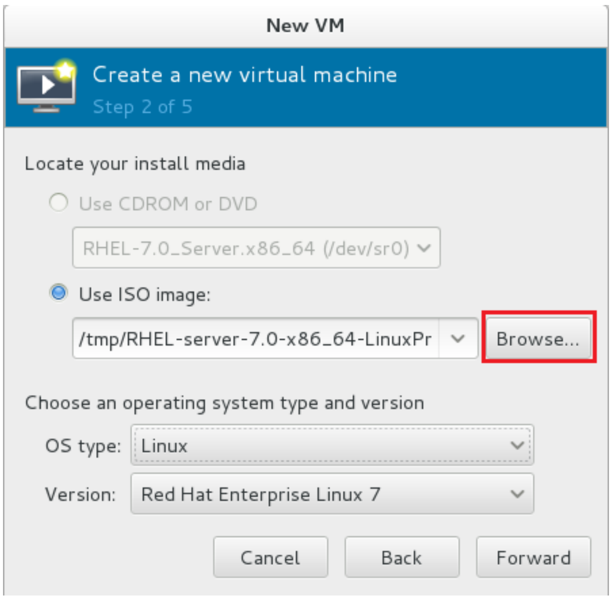
**太棒了！ 现在来配置虚拟机参数吧:**

[root@linuxprobe ~]# virt-manager

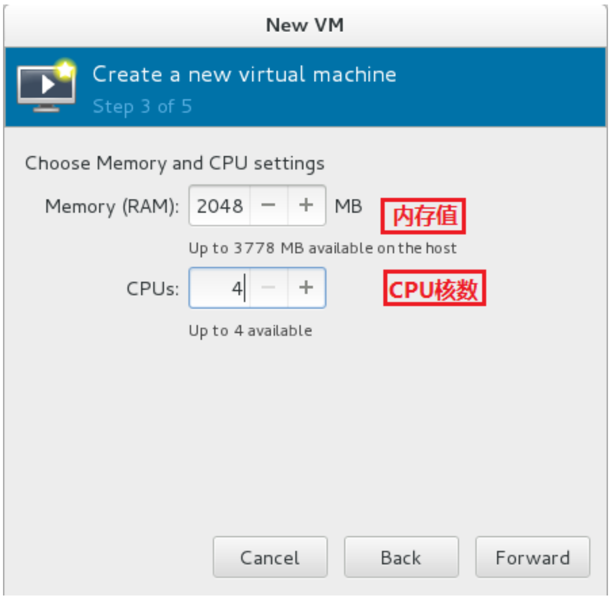
第 1 步：填写虚拟机名称和设置安装模式。



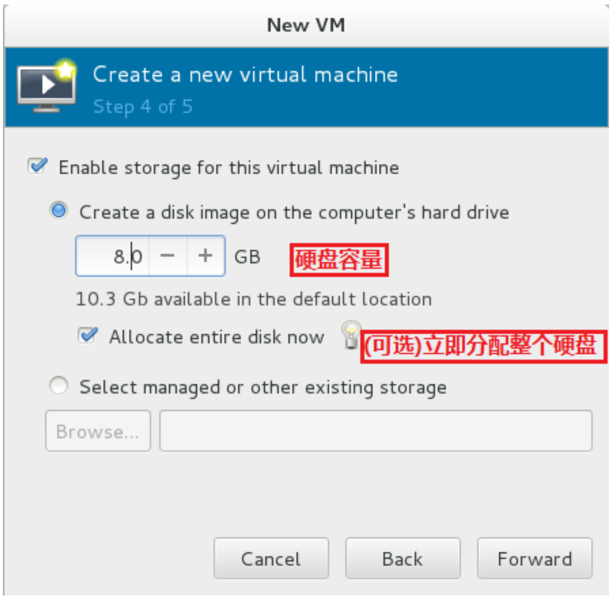

第 2 步：选中 RHEL7 镜像并设置系统类型。



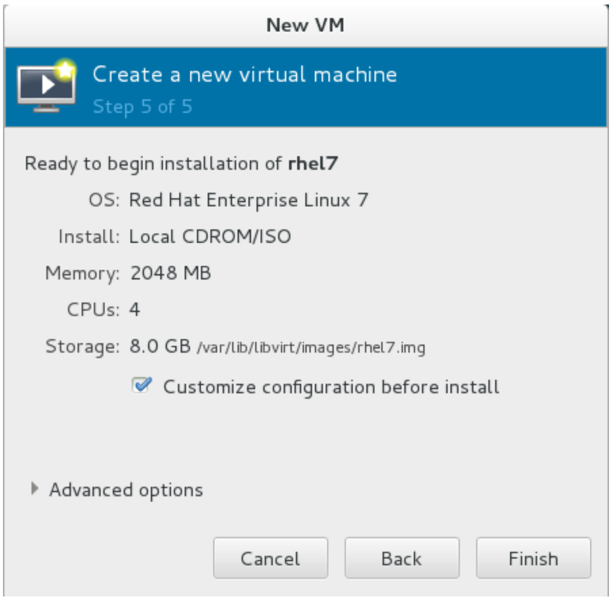
第 3 步：设置内存量与 CPU 核数。




第 4 步：定义硬盘容量。




第 5 步：检查配置后开启虚拟机。




1.4 配置 VNC 服务程序

**VNC 虚拟网络计算机**(Virtual Network Computing)是一款由欧洲实验室 AT&T 研发的远程控制程序，VNC 服务程序可以运行在类 Unix 计算机系统之上，拥有强大的远程控制能力，高效且实用。

如果您是 Linux 初学者或不需要远程控制功能，请先翻过本小节，等学完 Linux 命令了再来学~

在红帽 RHEL7 系统中 VNC 的服务软件包叫做 tigervnc-server：

[root@linuxprobe ~]# yum install tigervnc-server

Loaded plugins: langpacks, product-id, subscription-manager

………………省略部分安装过程………………

Installing:

tigervnc-server x86_64 1.2.80-0.30.20130314svn5065.el7 rhel7 199 k

………………省略部分安装过程………………

Complete!

复制一份 vnc 服务程序的配置文件(文件名中的:3 代表 5903 端口)：

[root@linuxprobe ~]# cp /lib/systemd/system/vncserver@.service /etc/systemd/system/vncserver@:3.service

编辑 vnc 服务的配置文件，将所有的修改为 linuxprobe 用户：

```
[root@linuxprobe ~]# vim /etc/systemd/system/vncserver@:3.service
[Unit]
Description=Remote desktop service (VNC)
After=syslog.target network.target
[Service]
Type=forking
# Clean any existing files in /tmp/.X11-unix environment
ExecStartPre=/bin/sh -c '/usr/bin/vncserver -kill %i > /dev/null 2>&1 || :'
ExecStart=/sbin/runuser -l <USER> -c "/usr/bin/vncserver %i"
PIDFile=/home/<USER>/.vnc/%H%i.pid
ExecStop=/bin/sh -c '/usr/bin/vncserver -kill %i > /dev/null 2>&1 || :'
[Install]
WantedBy=multi-user.target
```

配置防火墙规则：

```
[root@linuxprobe ~]# firewall-cmd --permanent --zone=public --add-port=5903/tcp
success
[root@linuxprobe ~]# firewall-cmd --reload
success
```

使用 linuxprobe 用户设置连接密码：

[linuxprobe@linuxprobe ~]$ vncserver

You will require a password to access your desktops.

Password:**此处输入连接密码**

Verify:**此处再次输入密码**

New 'linuxprobe.com:1 (linuxprobe)' desktop is linuxprobe.com:1

Creating default startup script /linuxprobe/.vnc/xstartup

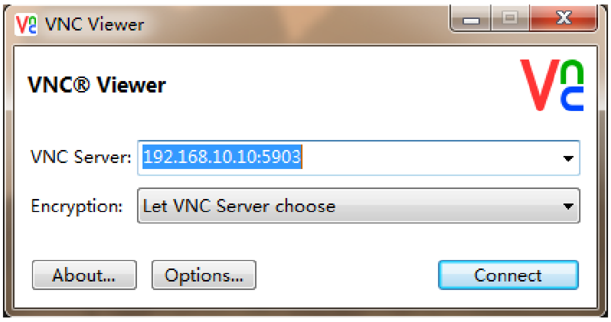
Starting applications specified in /linuxprobe/.vnc/xstartup

Log file is /linuxprobe/.vnc/linuxprobe.com:1.log

将 vncserver 服务程序启动并加入到开机启动项中：

```
[root@linuxprobe ~]# systemctl start vncserver@:3.service
[root@linuxprobe ~]# systemctl enable vncserver@:3.service
```

此时便可以使用 vnc 客户端工具连接啦



但是如果您出现了这样的报错：

xauth: (stdin):1: bad display name "linuxprobe.com:1" in "add" command

代表您的主机名(hostname)不能被 ping 通，请执行这行命令：

echo "127.0.0.1 linuxprobe.com" > /etc/hosts

### 1.5 重置 Root 用户密码

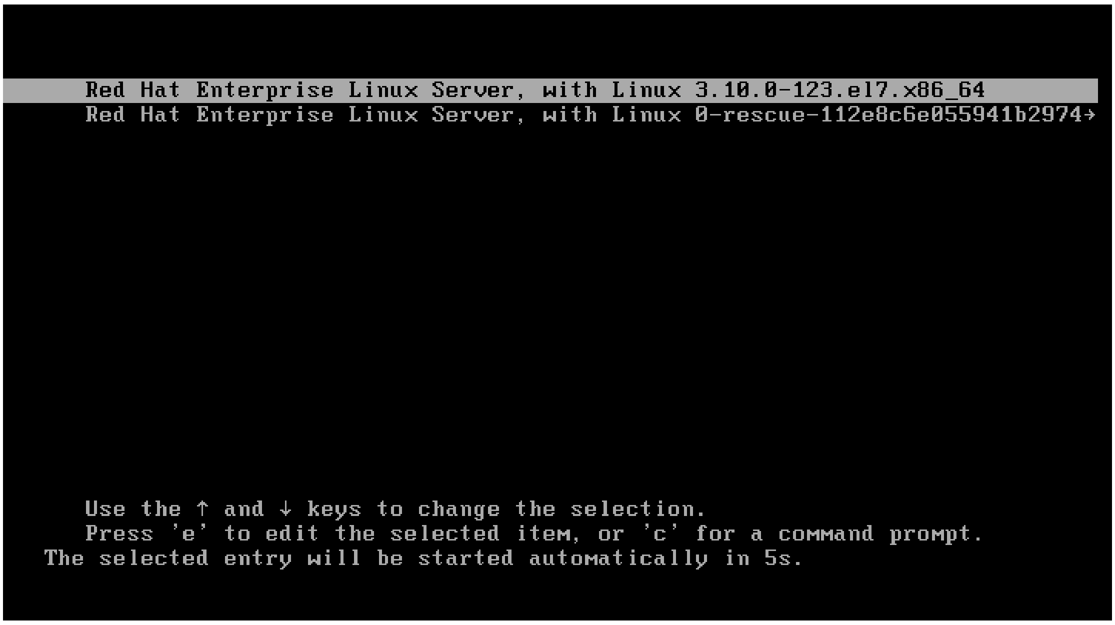
平日里让管理员很头疼的事情太多了，偶尔把密码忘记了也不用慌，重置密码只需简单几步，一定要学会哦！红帽 RHEL6 系统与红帽 RHEL7 系统破解系统密码方法完整版:http://www.linuxprobe.com/reset-root-password/

如果您是刚刚接手了一台 Linux 系统，请先确认这台系统是不是红帽 RHEL7 系统再进行下面的操作哦：

[root@linuxprobe ~]# cat /etc/redhat-release

Red Hat Enterprise Linux Server release 7.0 (Maipo)

第 1 步：开机后在内核上敲击“e”。




第 2 步：在 linux16 这行的后面输入“rd.break”并敲击“ctrl+x“。



第 3 步：进入到了系统的紧急求援模式。




第 4 步：依次输入以下命令。

```
mount -o remount,rw /sysroot
chroot /sysroot
echo "linuxprobe" | passwd --stdin root
touch /.autorelabel
exit
reboot
```




第 5 步：重启时会很慢，耐心等待即可。

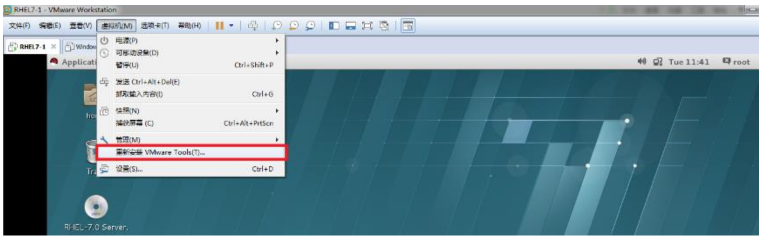



### 1.6 安装虚拟机增加包

VMware Tools 是 VMware 虚拟机中自带的增强工具包，用于增强虚拟机显卡与硬盘性能、同步虚拟机与主机的时钟时间、最主要的是可以支持虚拟机与主机之间的文件拖拽传输。

《Linux 就该这么学》的第二章才会正式接触 Linux 命令，**所以此刻您暂且无需对下面的安装过程完全理解。**

**第 1 步:在虚拟软件中选择“安装/重新安装 VMware Tools(T)”：**



**第 2 步:安装 VMwareTools 功能增加包（请用 root 用户登陆系统）：**

创建/media/cdrom 目录:

[root@linuxprobe ~]# mkdir -p /media/cdrom

将光驱设备挂载到该目录上：

[root@linuxprobe ~]# mount /dev/cdrom /media/cdrom

进入到该挂载目录：

[root@linuxprobe ~]# cd /media/cdrom

将功能增强包复制到/home 目录中：

[root@linuxprobe cdrom]# cp VMwareTools-9.9.0-2304977.tar.gz /home

进入到/home 目录中：

[root@linuxprobe cdrom]# cd /home

解压功能增强包：

root@linuxprobe home]# tar xzvf VMwareTools-9.9.0-2304977.tar.gz
vmware-tools-distrib/
vmware-tools-distrib/FILES
vmware-tools-distrib/doc/
vmware-tools-distrib/doc/open_source_licenses.txt
vmware-tools-distrib/doc/INSTALL
vmware-tools-distrib/doc/README
vmware-tools-distrib/installer/
vmware-tools-distrib/installer/services.sh
vmware-tools-distrib/installer/guestproxy-ssl.conf
vmware-tools-distrib/installer/thinprint.sh
vmware-tools-distrib/installer/upstart-job.conf
………………此处省略解压过程细节………………

进入解压文件夹中：

[root@linuxprobe home]# cd vmware-tools-distrib/

运行安装脚本并加上参数-d，代表默认安装：

[root@linuxprobe vmware-tools-distrib]# ./vmware-install.pl -d
The installer has detected an existing installation of open-vm-tools on this
system and will not attempt to remove and replace these user-space
applications. It is recommended to use the open-vm-tools packages provided by
the operating system. If you do not want to use the existing installation of
open-vm-tools and attempt to install VMware Tools, you must uninstall the
open-vm-tools packages and re-run this installer.
The installer will next check if there are any missing kernel drivers. Type yes
if you want to do this, otherwise type no [yes]
………………省略部分安装过程………………

当您看到这个字样后，重启后即可正常使用 VmwareTools 啦。

Creating a new initrd boot image for the kernel.
Starting Virtual Printing daemon: done
Starting vmware-tools (via systemctl): [ OK ]
The configuration of VMware Tools 9.9.0 build-2304977 for Linux for this
running kernel completed successfully.
Enjoy,
--the VMware team

**第 3 步：重新启动系统后生效：**

[root@linuxprobe ~]# reboot

### 1.7 重要的守护进程

当给一台主机安装上 Linux 系统后就可以工作了——包括接受用户的输入/计算/存储/再将结果输出等等，这是都是系统服务帮助我们完成的。而有一些系统服务需要时刻等待用户的输入（如键盘进程）或随时响应用户的请求（如网站服务进程）等等。

**守护进程（Daemon）**通常会随系统启动时激活并随系统关闭时停止，一直在系统后台中默默为用户提供服务：

| 守护进程名称 | 用处 |
| --- | --- |
| crond | 计划任务 |
| dhcpd | 动态 IP 地址分配服务（DHCP） |
| httpd | 网站服务 |
| lpd | 打印服务器 |
| named | 域名解析服务（DNS） |
| nfs | 文件共享服务（NFS） |
| smb | 文件共享与打印服务（SAMBA） |
| syslog | 系统日志 |
| gpm | 鼠标进程 |

### 1.8 红帽软件包管理器

在红帽软件包管理器（RPM）公布之前要想在 Linux 系统中安装软件只能采取"源码包"的方式安装，早期在 Linux 系统中安装程序是一件非常困难，耗费耐心的事情，因为大多数的服务程序仅仅提供编译源码，需要运维人员自行编译代码并解决许多的依赖关系，源码安装需要运维人员有很多的知识、高超的技能、甚至很好的耐心才能安装好一个程序，而且在安装、升级、卸载时还要考虑到其他程序、库的依赖关系，所以管理员在校验、安装、卸载、查询、升级等管理软件操作时难度非常大。

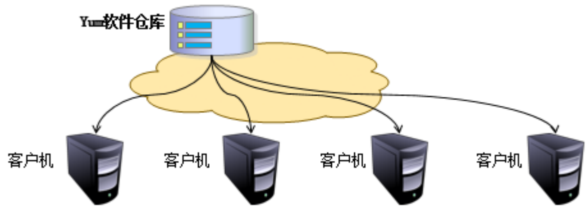
而 RPM 机制则为解决这些问题而设计的，RPM 原称为"Redhat Package Manager"，因其卓越的优势很快被公众认可，目前使用范围也已不局限在红帽系统中了。RPM 会建立统一的数据库文件，详细的记录软件信息并能够自动分析依赖关系，颇有一些"**软件控制面板**"的感觉。

**安装软件:**rpm -ivh filename.rpm
**升级软件:**rpm -Uvh filename.rpm
**卸载软件:**rpm -e filename.rpm
**查询软件的描述信息:**rpm -qpi filename.rpm
**列出软件的文件信息:**rpm -qpl filename.rpm
**查询文件属于那个 RPM:**rpm -qf filename

虽然 RPM 能够帮助用户查询软件相关的依赖关系，但问题还是要自己解决，有些大型软件需要数十个依赖包也是不小的负担。

1.9 Yum 软件仓库

Yum 仓库则是为进一步简化 RPM 管理软件难度而设计的，Yum 能够根据用户的要求分析出所需软件包及其相关依赖关系，自动从服务器下载软件包并安装到系统，听起来就已经很爽了吧?




**yum 软件仓库的使用拓扑图**

用户能够根据需求来指定 Yum 仓库与是否校验软件包，而这些只需几条关键词即可完成，现在来学习下配置的方法.所有 Yum 仓库的配置文件均需以.repo 结尾并存放在/etc/yum.repos.d/目录中的。

[rhel-media]: yum 源的名称，可自定义。

baseurl=file:///media/cdrom :提供方式包括 FTP（ftp://..)、HTTP（http://..)、本地（file:///..)

enabled=1 : 设置此源是否可用，1 为可用，0 为禁用。

gpgcheck=1 : 设置此源是否校验文件，1 为校验，0 为不校验。

gpgkey=file:///media/cdrom/RPM-GPG-KEY-redhat-release :若为校验请指定公钥文件地址。

Yum 仓库中的 RPM 软件包可以是由红帽官方发布的，也可以是第三方组织发布的，当然用户也可以编写的~

本书提供的镜像光盘内已经包含了大量的可用 RPM 软件包，将会在后面的实验章节中为大家演示如何使用。

| 命令 | 作用 |
| --- | --- |
| yum repolist all | 列出所有仓库。 |
| yum list all | 列出仓库中所有软件包 |
| yum info 软件包名称 | 查看软件包信息 |
| yum install 软件包名称 | 安装软件包 |
| yum reinstall 软件包名称 | 重新安装软件包 |
| yum update 软件包名称 | 升级软件包 |
| yum remove 软件包 | 移除软件包 |
| yum clean alla | 清除所有仓库缓存 |
| yum check-update | 检查可更新的软件包 |
| yum grouplist | 查看系统中已经安装的软件包组 |
| yum groupinstall 软件包组 | 安装指定的软件包组 |
| yum groupremove 软件包组 | 移除指定的软件包组 |
| yum groupinfo 软件包组 | 查询指定的软件包组信息 |

第 2 章 新手必须掌握的 Linux 命令。

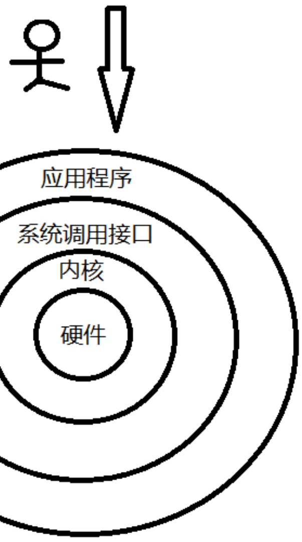
**章节简述：**

本章节讲述系统内核、Bash 解释器的关系与作用，教给读者如何正确的执行 Linux 命令以及常见排错方法。
经验丰富的运维人员可以恰当的组合命令与参数，使 Linux 字符命令更加的灵活且相对减少消耗系统资源。
已经收录了上百个最常用的 Linux 命令，其中有数十个命令被放到了后面的章节，到时候咱们再随用随学~

## 2.1 强大好用的 SHELL

计算机硬件是由**运算器**、**控制器**、**存储器**、**输入/输出设备**等设备组成的，而能够让机箱内各种设备各司其职东西就叫做——**系统内核**。内核负责驱动硬件、管理活动和分配/管理硬件资源，如此说来系统内核对计算机来讲可真的是太重要了，**所以它不能直接让用户操作。**

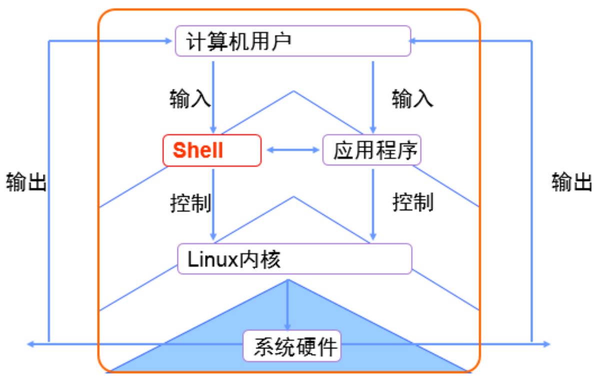
因为用户不能直接控制硬件也不能直接操作内核，于是便需要基于"系统调用接口"开发出的程序/服务来满足用户日常工作了。




首先承认在红帽 RHEL7 中有些诸如逻辑卷管理器（LVM）的图形化工具非常好用，也减少了运维人员操作出错的几率，值得称赞，但一直以来 Linux 运维人员更多的倾向于用命令写脚本程序，**因为图形化的工具不灵活而且相比来说更加消耗系统资源。**

其实很多图形工具也是调用脚本来工作的，但功能却被"阉割"了，更缺乏了灵活性，所以有些运维人员甚至都不会给 Linux 系统安装图形界面，需要工作了直接远程连接过去，不得不说这样做真的挺高效的。

"Shell"——也可称为"壳"，**充当的是人与内核（硬件）的翻译官**，用户将一些命令"告诉"Shell，它就会调用相应的程序服务执行工作啦，很厉害吧~~现在包括红帽系统在内的许多热门 Linux 系统主流默认字符 Shell 是 **Bash(Bourne-Again SHell)。**




读者要明白 bash 作为大多数 linux 系统的默认字符解释器，必须必须必须得学好！Bash 的优势：

1.默认保存历史命令（可用上下键翻看）

2.命令仅需输入前几位就可以用 tab 键补全(RHEL7 更牛的是参数补全)

3.强大的批处理脚本

4.实用的环境变量

## 2.2 执行命令与查看帮助

既然有了如此好用的“翻译官”，那么接下来就有必要好好学习下如何更高效的和它沟通了~

要想准确的、高效的完成工作，不能够光靠命令本身，还应该根据实际情况来组合各种命令选择和命令参数：

**命令名称 [命令参数] [命令对象]**

注意:命令名称、命令参数、命令对象之间请用空格键分隔。

比较好理解的是命令对象，**命令对象一般是指要处理的目标（普通文件/目录文件/用户等等）**，而命令参数对于新手来讲比较麻烦，因为这个值会随命令的不同和环境情况的不同而异，所以在参数选择搭配上需要长时间的经验积累才可以。

命令的参数可以选用**长格式**(完整的选项名称)也可选用**短格式**(单个字母的缩写)，分别用”—“与” - “做前缀。

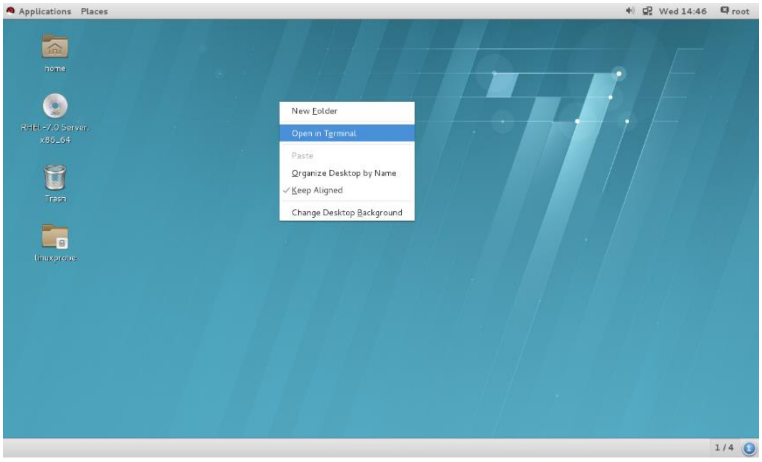
长格式如:man －**help**

短格式如:man -**h**

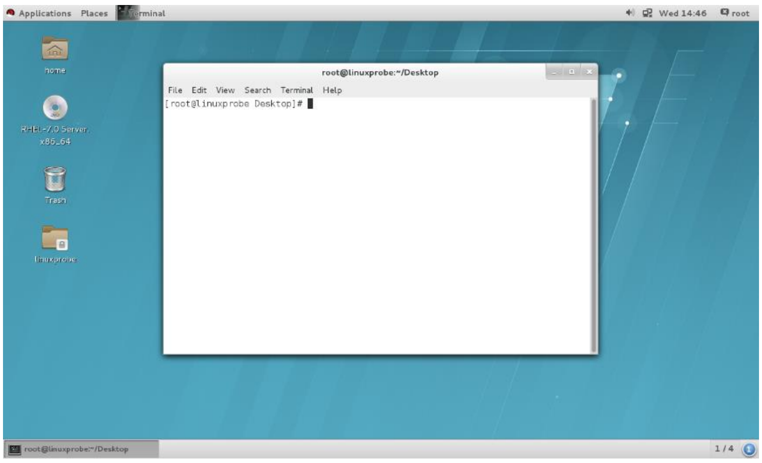
当遇到了一个陌生命令后如何知道它有那些可用的参数？这时就可以用 man 命令了。

本书将 man 命令作为第一个要学的 Linux 命令是因为它的作用非常强大——**可用于查看命令的具体可用参数与对象格式等等。**

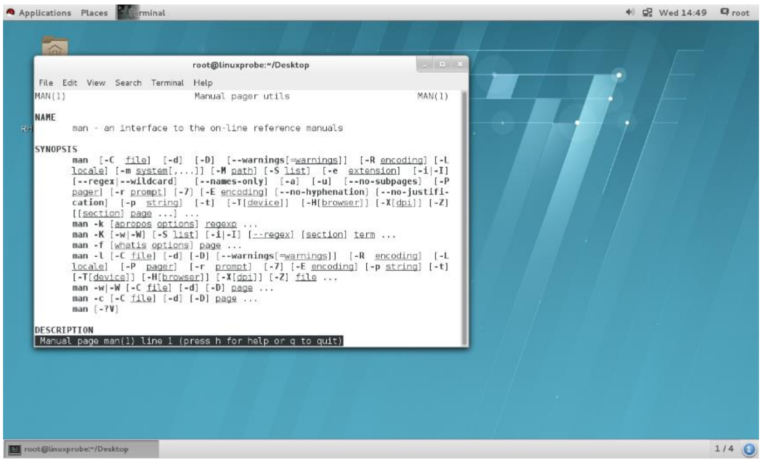
运行虚拟机中的 RHEL7 系统，并在桌面上敲击右键后点击“Open in Terminal”，这样就成功的打开了一个终端。




输入字符“man man”来用 man 命令查看自身的帮助信息。



man 命令的帮助信息界面




**man 命令的可用帮助文档分类有：**

| 代码 | 代表内容 |
| --- | --- |
| 1 | 普通的命令 |
| 2 | 内核调用的函数与工具 |
| 3 | 常见的函数与函数库 |
| 4 | 设备文件的说明 |
| 5 | 配置文件 |
| 6 | 游戏 |
| 7 | 惯例与协议 |
| 8 | 管理员可用的命令 |
| 9 | 内核相关的文件 |

**一般来讲帮助文件都很长很多，如果你想翻看的话，就要理解帮助文档的目录结构与操作方法：**

| 结构名称 | 代表意义 |
| --- | --- |
| NAME | 命令的名称 |
| SYNOPSYS | 参数的大致使用方法 |
| DESCRIPTION | 介绍说明 |
| EXAMPLES | 演示（附带简单说明） |
| OVERVIEW | 概述 |

| | |
|---|---|
| DEFAULTS | 默认的功能 |
| OPTIONS | 具体的可用选项（带介绍） |
| ENVIRONMENT | 环境变量 |
| FILES | 用到的文件 |
| SEE ALSO | 相关的资料 |
| HISTORY | 维护历史与联系方式 |

**man 命令的操作按键:**

| 按键 | 用处 |
|---|---|
| 空格键 | 向下翻一页。 |
| [Page Down] | 向下翻一页。 |
| [Page Up] | 向上翻一页。 |
| [HOME] | 直接前往首页。 |
| [END] | 直接前往尾页。 |
| /关键词 | 从上至下搜索某个关键词,如"/linux"。 |
| ?关键词 | 从下至上搜索某个关键词,如"?linux"。 |
| n | 定位到下一个搜索到的关键词。 |
| N | 定位到上一个搜索到的关键词。 |
| q | 退出帮助文档。 |

### 2.3 常用系统工作命令

刚刚学会了一个重量级的 man 命令，感觉很不错吧？接下来就是常用的命令啦，尽量背记下来，当然实在不行回来查也可以的，**echo 命令用于在终端显示字符串或变量，格式为：“echo [字符串 | 变量]”。**

将 echo 命令的字符串输出到终端:

[root@linuxprobe ~]# echo Linuxprobe.Com

Linuxprobe.Com

用 echo 命令查看 SHELL 变量的值（前面有$符号）:

[root@linuxprobe ~]# echo $SHELL

/bin/bash

查看本机主机名：

[root@linuxprobe ~]# echo $HOSTNAME

Linuxprobe.Com

**date 命令用于显示/设置系统的时间或日期，格式为：”date [选项] [+指定的格式]”。**

强大的 date 命令能够按照指定格式显示系统的时间或日期，只需键入”+”号开头的字符串指定其格式，详细格式如下：

| 参数 | 作用 |
|---|---|
| %t | 跳格[TAB 键] |
| %H | 小时(00-23) |
| %I | 小时(01-12) |
| %M | 分钟(00-59) |
| %S | 秒（00-60） |
| %X | 相当于%H:%M:%S |
| %Z | 显示时区 |
| %p | 显示本地 AM 或 PM |
| %A | 星期几 (Sunday-Saturday) |
| %a | 星期几 (Sun-Sat) |
| %B | 完整月份 (January-December) |
| %b | 缩写月份 (Jan-Dec) |
| %d | 日(01-31) |
| %j | 一年中的第几天(001-366) |
| %m | 月份(01-12) |
| %Y | 完整的年份 |

查看当前的系统时间：

[root@linuxprobe ~]# date

Mon Aug 24 16:11:23 CST 2015

按照”年-月-日 小时:分钟:秒”的格式：

[root@linuxprobe ~]# date "+%Y-%m-%d %H:%M:%S"

2015-08-24 16:29:12

设置系统时间为 2015 年 9 月 1 日 8 点半：

[root@linuxprobe ~]# date -s "20150901 8:30:00"

Tue Sep 1 08:30:00 CST 2015

查看当前系统时间：

[root@linuxprobe ~]# date

Tue Sep 1 08:30:01 CST 2015

查看本地系统时区：

[root@linuxprobe ~]# date "+%Z"

CST

查看星期几：

[root@linuxprobe ~]# date "+%A"

Tuesday

输入当前是上午还是下午：

[root@linuxprobe Desktop]# date "+%p"

AM

判断今天是一年中的第几天：

[root@linuxprobe ~]# date "+%j"

244

**reboot 命令用于重启系统(仅 root 用户可以使用)，格式为：”reboot”。**

重启计算机：

[root@linuxprobe ~]# reboot

**wget 命令用于使用命令行下载网络文件，格式为：“wget [参数] 下载地址”。**

| 参数 | 作用 |
|---|---|
| -b | 后台下载模式。 |
| -O | 下载到指定目录。 |
| -t | 最大尝试次数。 |
| -c | 断点续传 |
| -p | 下载页面内所有资源,包括图片、视频等。 |
| -r | 递归下载 |

首先需要配置您的 Linux 系统能够正常登入互联网，然后使用 wget 命令下载由《Linux 就该这么学》提供的红帽 RHEL7 系统镜像：

[root@linuxprobe ~]# *wget http://www.linuxprobe.com/Tools/RHEL-server-7.0-x86_64-LinuxProbe.Com.iso*

--2015-09-01 18:25:24-- http://www.linuxprobe.com/Tools/RHEL-server-7.0-x86_64-LinuxProbe.Com.iso

Resolving www.linuxprobe.com... 106.185.25.197

Connecting to www.linuxprobe.com|106.185.25.197|:80... connected.

Saving to: 'RHEL-server-7.0-x86_64-LinuxProbe.Com.iso'

100%[=====================================>] 3,743,416,320 1.82M/s in 32m 27s

2015-09-01 18:57:51 (1.83 MB/s) - 'RHEL-server-7.0-x86_64-LinuxProbe.Com.iso' saved [3743416320/3743416320]

递归下载《Linux 就该这么学》的整站页面与所有资料，下载完成后会在当前目录中保存成名为”www.linuxprobe.com“的目录：

[root@linuxprobe ~]# wget -r -p http://www.linuxprobe.com

--2015-09-01 18:31:41-- http://www.linuxprobe.com/

Resolving www.linuxprobe.com... 106.185.25.197

Connecting to www.linuxprobe.com|106.185.25.197|:80... connected.

HTTP request sent, awaiting response... 200 OK

Length: unspecified [text/html]

Saving to: ‘www.linuxprobe.com/index.html’

………………省略下载过程………………

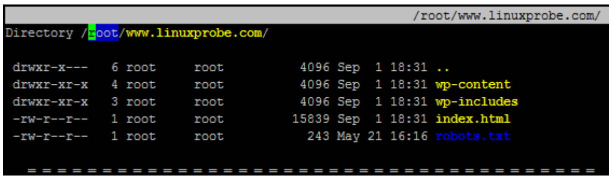
**elinks 用于实现一个纯文本界面的浏览器，格式为：“elinks [参数] 网址”。**

安装 elinks 纯文本浏览器：

[root@linuxprobe ~]# yum install elinks

使用 elinks 访问《Linux 就该这么学》：

[root@linuxprobe ~]# elinks www.linuxprobe.com




## 2.4 系统状态检测命令

合格的运维人员必需具备快速查看系统状态的能力，所以这些命令真的很常用呢！

**ifconfig 用于获取网卡配置与网络状态等信息:格式为”ifconfig [网络设备] [参数]”。**

查看本机当前的网卡配置与网络状态等信息：

[root@linuxprobe ~]# ifconfig

eno16777728: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 192.168.10.10 netmask 255.255.255.0 broadcast 192.168.10.255

inet6 fe80::20c:29ff:fe9c:6373 prefixlen 64 scopeid 0x20<link>

ether 00:0c:29:9c:63:73 txqueuelen 1000 (Ethernet)

RX packets 61 bytes 6612 (6.4 KiB)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 32 bytes 4511 (4.4 KiB)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536

inet 127.0.0.1 netmask 255.0.0.0

inet6 ::1 prefixlen 128 scopeid 0x10<host>

loop txqueuelen 0 (Local Loopback)

RX packets 2 bytes 140 (140.0 B)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 2 bytes 140 (140.0 B)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

**uname 命令用于查看系统内核版本等信息，格式为："uname [-a]"。**

查看系统的内核名称、内核发行版、内核版本、节点名、硬件名称、硬件平台、处理器类型、操作系统等信息:

[root@linuxprobe ~]# uname -a

Linux linuxprobe.com 3.10.0-123.el7.x86_64 #1 SMP Mon May 5 11:16:57 EDT 2014 x86_64 x86_64 x86_64 GNU/Linux

顺便说下，如果您想查看系统详细版本信息就看 redhat-release 文件:

[root@linuxprobe ~]# cat /etc/redhat-release

Red Hat Enterprise Linux Server release 7.0 (Maipo)

**uptime 命令用于查看系统的负载情况，格式为："uptime"。**

我也经常用"**watch -n 1 uptime**"来每秒刷新一次获得当前的系统负载情况，输出内容分别为系统当前时间、系统已运行时间、当前在线用户以及平均负载值。而平均负载分为最近 1 分钟、5 分钟、15 分钟的系统负载情况，负载值越低越好(小于 1 是正常)。

获取当前系统状态信息:

[root@linuxprobe ~]# uptime

22:49:55 up 10 min, 2 users, load average: 0.01, 0.19, 0.18

**free 命令用于显示当前系统中内存的使用量情况，格式为："free [-m/-g]"。**

以 m 为单位显示当前系统中内存的使用量情况:

[root@linuxprobe ~]# free -m

| | 总计内存量 | 已用量 | 可用量 | 进程共享的内存量 | 磁盘缓存的内存量 | 缓存的内存量 |
|---|---|---|---|---|---|---|
| | total | used | free | shared | buffers | cached |
| Mem: | 1483 | 885 | 598 | 9 | 0 | 255 |
| -/+ buffers/cache: | | 628 | 855 | | | |
| Swap: | 2047 | 0 | 2047 | | | |

**who 命令用于查看当前登入主机的用户情况，格式为："who [参数]"。**

查看当前登入主机用户的情况:

[root@linuxprobe ~]# who

| 登陆的用户名 | 终端设备 | 登陆到系统的时间 |
|---|---|---|
| root | :0 | 2015-08-24 17:52 (:0) |
| root | pts/0 | 2015-08-24 17:52 (:0) |

**last 命令用于查看所有系统的登入记录，格式为："last [参数]"。**

查看系统的登入记录:

[root@linuxprobe ~]# last

root pts/0 :0 Mon Aug 24 17:52 still logged in

root :0 :0 Mon Aug 24 17:52 still logged in

(unknown :0 :0 Mon Aug 24 17:50 - 17:52 (00:02)

reboot system boot 3.10.0-123.el7.x Tue Aug 25 01:49 - 18:17 (-7:-32)

root pts/0 :0 Mon Aug 24 15:40 - 08:54 (7+17:14)

root pts/0 :0 Fri Jul 10 10:49 - 15:37 (45+04:47)

**history 命令用于显示历史执行过的命令，格式为："history [-c]"。**

查看当前用户在系统中执行过的命令：

[root@linuxprobe ~]# history

1 tar xzvf VMwareTools-9.9.0-2304977.tar.gz

2 cd vmware-tools-distrib/

3 ls

4 ./vmware-install.pl -d

5 reboot

6 df -h

7 cd /run/media/

8 ls

9 cd root/

10 ls

11 cd VMware\ Tools/

12 ls

13 cp VMwareTools-9.9.0-2304977.tar.gz /home

14 cd /home

15 ls

16 tar xzvf VMwareTools-9.9.0-2304977.tar.gz

17 cd vmware-tools-distrib/

18 ls

19 ./vmware-install.pl -d

20 reboot

21 history

历史命令会被保存到用户家目录中的".bash_history"文件中。Linux 系统中以点(.)开头的文件均代表隐藏文件，一般会是系统文件。

[root@linuxprobe ~]# cat ~/.bash_history

清空该用户在本机中执行过命令的历史记录：

[root@linuxprobe ~]# history -c

history 默认会保存 1000 条执行过的命令，若要修改可直接编辑/etc/profile 文件的 HISTSIZE 值。

**sosreport 命令用于收集系统系统配置并诊断信息后输出结论文档，格式为："sosreport"。**

当我们的红帽系统出现故障需要联系红帽厂商或其他技术支持时，大多数情况都需要提供使用到这个命令。

收集系统本地配置信息并诊断：

[root@linuxprobe ~]# sosreport

### 2.5 工作目录切换命令

虽然后面的章节才会学习到 linux 的存储结构与"目录"这个概念，但没有关系，现在您也能看懂下面的操作。

**pwd 命令用于显示当前的工作目录，格式为："pwd [选项]"。**

| 参数 | 作用 |
| --- | --- |
| -P | 显示真实路径。(即非快捷链接的地址) |

查看当前的工作路径：

[root@linuxprobe etc]# pwd

/etc

**cd 命令用于切换工作路径，格式为："cd [目录名称]"。**

| 参数 | 作用 |
| --- | --- |
| - | 切换到上一次的目录，如"cd -" |
| ~ | 切换到"家目录"，如"cd ~" |
| ~username | 切换到其他用户的家目录，如"cd ~teak" |
| .. | 切换到上级目录，如"cd .." |

切换进/etc 目录中：

[root@linuxprobe ~]# cd /etc

切换进/bin 目录中：

[root@linuxprobe etc]# cd /bin

返回上级目录（即/etc 目录）：

[root@linuxprobe bin]# cd -

/etc

返回用户自己的家目录：

[root@linuxprobe etc]# cd ~

[root@linuxprobe ~]#

**ls 命令用于查看目录中有那些文件，格式为："ls [选项] [文件] "。**

查看当前目录下有那些文件（长格式）：

[root@linuxprobe etc]# ls -al

| 参数 | 作用 |
| --- | --- |
| -a | 查看全部文件（包括隐藏文件） |
| -d | 仅看目录本身 |
| -h | 易读的文件容量（如 k,m,g） |
| -l | 显示文件的详细信息 |

查看/etc 目录中有那些文件：

[root@linuxprobe ~]# ls /etc

abrt gss printcap

adjtime gssproxy profile

aliases gtk-2.0 profile.d

aliases.db gtk-3.0 protocols

alsa hba.conf pulse

alternatives host.conf purple

anacrontab hostname qemu-ga

asound.conf hosts qemu-kvm

at.deny hosts.allow radvd.conf

………省略部分文件………

查看/etc 目录的权限与属性：

[root@linuxprobe ~]# ls -ld /etc

drwxr-xr-x. 132 root root 8192 Jul 10 10:48 /etc

追加-h 参数，以 K/M/G 为单位显示容量：

[root@linuxprobe ~]# ls -ldh /etc

drwxr-xr-x. 132 root root 8.0K Jul 10 10:48 /etc

**2.6 文本文件编辑命令**

既然已经学会了工作目录间的切换与查看，那么就来试试对文件的一系列操作吧，非常实用。

**cat 命令用于查看纯文本文件（较短的），格式为："cat [选项] [文件]"。**

查看文本文件：

[root@linuxprobe ~]# cat 文件名

| 参数 | 作用 |
| --- | --- |
| -n | 显示行号 |
| -b | 显示行号（不包括空行） |
| -A | 显示出"不可见"的符号，如空格，tab 键等等 |

**more 命令用于查看纯文本文件（较长的），格式为："more [选项] 文件"。**

查看文本文件：

[root@linuxprobe ~]# more 文件名

| 参数 | 作用 |
| --- | --- |
| -数字 | 预先显示的行数（默认为一页） |
| -d | 显示提示语句与报错信息 |

**head 命令用于查看纯文本文档的前 N 行，格式为："head [选项] [文件]"。**

查看文本文件前 20 行：

[root@linuxprobe ~]# head -n 20 文件名

| 参数 | 作用 |
| --- | --- |
| -n 10 | 显示 10 行 |
| -n -10 | 正常输出（如 cat 命令），但不显示最后的 10 行 |

**tail 命令用于查看纯文本文档的后 N 行，格式为："tail [选项] [文件]"。**

查看文本文件后 20 行：

[root@linuxprobe ~]# tail -n 20 文件名

| 参数 | 作用 |
| --- | --- |
| -n 10 | 显示后面的 10 行 |
| -f | 持续刷新显示的内容 |

**od 命令用于对查看特殊格式的文件，格式为：“od [选项] [文件]”。**

| 参数 | 作用 |
| --- | --- |
| -t a | 默认字符 |
| -t c | ASCII 字符 |
| -t o | 八进制 |
| -t d | 十进制 |
| -t x | 十六进制 |
| -t f | 浮点数 |

**tr 命令用于转换文本文件中的字符，格式为:“tr [原始字符] [目标字符]”。**

读者如果想转换实例中的文件，可下载文件 tr.txt。

将 tr.txt 文件的内容转换成大写（注意到命令中间的|了吗？这个叫管道命令符，后面小节会学习到的）

[root@linuxprobe ~]# cat tr.txt | tr [a-z] [A-Z]

WELCOME TO LINUXPROBE.COM

RED HAT CERTIFIED

FREE LINUX LESSONS

PROFESSIONAL GUIDANCE

LINUX COURSE

**wc 命令用于统计指定文本的行数、字数、字节数，格式为“wc [参数] 文本”。**

| 参数 | 作用 |
| --- | --- |
| -l | 只显示行数 |
| -w | 只显示单词数 |
| -c | 只显示字节数 |

统计当前系统中的用户个数：

[root@linuxprobe ~]# wc -l /etc/passwd

38 /etc/passwd

**cut 命令用于通过列来提取文本字符，格式为：“cut [参数] 文本”。**

| 参数 | 作用 |
| --- | --- |
| -d 分隔符 | 指定分隔符，默认为 Tab。 |
| -f | 指定显示的列数。 |
| -c | 单位改为字符 |

获取当前系统中所有用户的名称:
参数作用: -d 以”:”来做分隔符，-f 参数代表只看第一列的内容。
[root@linuxprobe ~]# cut -d: -f1 /etc/passwd
获取 root 用户的默认 SHELL 解释器:
[root@linuxprobe ~]# grep ^root /etc/passwd | cut -d: -f 7
/bin/bash
**diff 命令用于比较多个文本文件的差异，格式为：”diff [参数] 文件”。**
读者如果想比较实例中的文件，可点此下载文件 diff_A.txt 与 diff_B.txt。

| 参数 | 命令 |
| --- | --- |
| -b | 忽略空格引起的差异。 |
| -B | 忽略空行引起的差异。 |
| --brief 或-q | 仅报告是否存在差异。 |
| -c | 使用上下文输出格式。 |

比较两个文件的差异:
[root@linuxprobe ~]# diff diff_A.txt diff_B.txt
1c1,2
> Welcome to linuxprobe.com
---
> Welcome tooo linuxprobe.com
>
3c4,5
> Free Linux Lessons
---
> Free Linux LeSSonS
> ////////...../////////
仅显示比较后的结果，即相同或不相同:
[root@linuxprobe ~]# diff --brief diff_A.txt diff_B.txt
Files diff_A.txt and diff_B.txt differ
使用上下文输出的格式:
[root@linuxprobe ~]# diff -c diff_A.txt diff_B.txt
*** diff_A.txt 2015-08-30 18:07:45.230864626 +0800
--- diff_B.txt 2015-08-30 18:08:52.203860389 +0800
***************
*** 1,5 ****
! Welcome to linuxprobe.com
Red Hat certified
! Free Linux Lessons
Professional guidance
Linux Course
--- 1,7 ----

! Welcome tooo linuxprobe.com
!
Red Hat certified
! Free Linux LeSSonS
! ////////...../////////
Professional guidance
Linux Course

**2.7 文件目录管理命令**

**touch 命令用于创建空白文件与修改文件时间，格式为：“touch [选项] [文件]”。**

我们可以用“**touch test**”轻松的创建出一个名字为 test 的空白文档，所以这个功能无须介绍。

**对于在 Linux 中的文件有三种时间：**

更改时间(mtime):内容修改时间（不包括权限的）
更改权限(ctime):更改权限与属性的时间
读取时间(atime):读取文件内容的时间

如果黑客执行了 **touch** -d “2 days ago” test，便将访问与修改时间修改为了 2 天前（**伪造了自己没有动过该文件的假象**）。

| 参数 | 作用 |
|---|---|
| -a | 近修改“访问时间”（atime） |
| -m | 近修改“更改时间”（mtime） |
| -d | 同时修改 atime 与 mtime |
| -t | 要修改成的时间[YYMMDDhhmm] |

**mkdir 用于创建空白的文件夹，格式为：“mkdir [选项] 目录”。**

创建文件夹：

[root@linuxprobe ~]# mkdir 文件夹名

| 参数 | 作用 |
|---|---|
| -m=MODE | 默认的文件目录权限，如"-m 755" |
| -p | 连续创建多层目录（若文件夹已存在则忽略） |
| -v | 显示创建的过程 |

创建一个名字叫 linuxprobe 的目录：

[root@linuxprobe ~]# mkdir linuxprobe

使用 ls 命令查看该目录的权限属性等信息：

[root@linuxprobe ~]# ls -ld linuxprobe/

drwxr-xr-x. 2 root root 6 Aug 24 19:25 linuxprobe/

还记得刚刚用 cd 命令进入 linuxprobe 目录吗？这里是个小技巧，变量!$或(键盘按键)代表上一条命令的参数。

[root@linuxprobe ~]# cd !$

cd linuxprobe

pwd 命令也是刚刚学习过的，用于显示当前的工作路径。

[root@linuxprobe linuxprobe]# pwd

/root/Desktop/linuxprobe

一次创建 5 个目录 a/b/c/d/e：

[root@linuxprobe linuxprobe]# mkdir -p a/b/c/d/e

查看目录的属性，验证是否成功：

[root@linuxprobe linuxprobe]# ls -ld a/b/c/d/e/

drwxr-xr-x. 2 root root 6 Aug 29 10:16 a/b/c/d/e/

**cp 命令用于复制文件或目录，格式为："cp [选项] 源文件 目标文件"。**

复制命令的三种情况:

目标文件是一个目录，会将源文件复制到该目录中。

目标文件是一个文件，会将源文件覆盖该文件。

目标文件不存在，将会复制源文件并修改为目标文件的名称（重命名）。

| 参数 | 作用 |
|---|---|
| -p | 保留原始文件的属性 |
| -d | 若对象为"链接文件"，则保留该"链接文件"的属性 |
| -r | 递归持续复制（用于目录） |
| -i | 若目标文件存在则询问是否覆盖 |
| -a | 相当于-pdr（p,d,r 为上述的参数） |

创建一个名为 install.log 的文件：

[root@linuxprobe ~]# touch install.log

将 install.log 复制为 x.log：

[root@linuxprobe ~]# cp install.log x.log

查看到确实出现了文件 x.log

[root@linuxprobe ~]# ls

install.log x.log

**mv 命令用于移动文件或改名，格式为："mv [选项] 文件名 [目标路径|目标文件名]"。**

将文件 aaa 重命名为 bbb：

[root@linuxprobe ~]# mv aaa bbb

**rm 命令用于删除文件或目录，格式为："rm [选项] 文件"。**

删除普通文件并提示确认信息:"rm 文件名"

删除普通文件或目录文件，不提示:"rm -rf 文件或目录名"

| 参数 | 作用 |
|---|---|
| -f | 忽略警告信息 |
| -i | 删除前先询问 |
| -r | 删除文件夹 |

查看当前目录下的文件：
[root@linuxprobe ~]# ls
install.log x.log
删除 install.log 文件，输入"y"即确认：
[root@linuxprobe ~]# rm install.log
rm: remove regular empty file 'install.log'? y
删除 x.log 文件而无需确认：
[root@linuxprobe ~]# rm -rf x.log
Linux 系统中还有一个 rmdir 命令，它不同于 rm -rf 命令会删除一切，而是仅删除空目录，遇到目录内有文件时则报错。
**dd 命令用于指定大小的拷贝的文件或指定转换文件，格式为："dd [参数]"。**

| 参数 | 作用 |
| --- | --- |
| if | 输入的文件名称。 |
| of | 输出的文件名称。 |
| bs | 设置每个"块"的大小。 |
| count | 设置要拷贝"块"的个数。 |
| conv=ucase | 将字母从小写转换为大写。 |
| conv=lcase | 把字符从大写转换为小写。 |

将光驱设备拷贝成镜像文件：
[root@linuxprobe ~]# dd if=/dev/cdrom of=RHEL-server-7.0-x86_64-LinuxProbe.Com.iso
7311360+0 records in
7311360+0 records out
3743416320 bytes (3.7 GB) copied, 370.758 s, 10.1 MB/s
生成一个 560m 的空白文件：
[root@linuxprobe ~]# dd if=/dev/zero of=560_file count=1 bs=560M
1+0 records in
1+0 records out
587202560 bytes (587 MB) copied, 27.1755 s, 21.6 MB/s
将硬盘的 MBR 信息拷贝出来：
[root@linuxprobe ~]# dd if=/dev/sda of=sda_image count=1 bs=512K
1+0 records in
1+0 records out
524288 bytes (524 kB) copied, 0.0449481 s, 11.7 MB/s

## 2.8 用户与组管理命令

**useradd 命令用于创建新的用户，格式为：“useradd [选项] 用户名”。**

| 参数 | 作用 |
|---|---|
| -d | 指定用户的家目录（默认为/home/username） |
| -D | 展示默认值 |
| -e | 帐号有效截至日期，格式：YYYY-MM-DD. |
| -g | 指定一个初始用户组（必须已存在） |
| -G | 指定一个或多个扩展用户组 |
| -N | 不创建与用户同名的用户组 |
| -s | 指定默认的 Shell |
| -u | 指定用户的 UID |

创建名为 linuxprobe 的用户，并定义家目录路径、UID 以及登陆解释器（不允许登陆）:

[root@linuxprobe ~]# useradd -d /home/linux -u 8888 -s /sbin/nologin linuxprobe

查看 linuxprobe 用户的基本信息:

[root@linuxprobe ~]# id linuxprobe

uid=8888(linuxprobe) gid=8888(linuxprobe) groups=8888(linuxprobe)

**passwd 命令用于修改用户的密码，格式为：“passwd [选项] [用户名]”。**

修改当前用户的密码:“passwd”

修改其他用户的密码:“passwd 其他用户名”

| 参数 | 作用 |
|---|---|
| -l | 锁定用户禁止其登陆 |
| -u | 解除锁定，允许用户登陆。 |
| --stdin | 允许从标准输入修改用户密码，如(echo "NewPassWord" \| passwd -stdin Username) |
| -d | 使帐号无密码 |
| -e | 强制用户下次登陆时修改密码 |
| -S | 显示用户的密码状态 |

**userdel 命令用于删除用户所有表格，格式为：“userdel [选项] 用户名”。**

删除用户与其家目录：

[root@linuxprobe ~]# userdel -d 用户名

| 参数 | 作用 |
| --- | --- |
| -f | 强制删除用户，家目录与其相关文件 |
| -r | 同时删除用户，家目录与其相关文件 |

**usermod 命令用于修改用户的属性，格式为“usermod [选项] 用户名”。**

| 参数 | 作用 |
| --- | --- |
| -c | 填写帐号的备注信息 |
| -d -m | -m 与-d 连用，可重新指定用户的家目录并自动旧的数据转移过去。 |
| -e | 帐户到期时间，格式“YYYY-MM-DD” |
| -g | 变更所属用户组 |
| -G | 变更扩展用户组 |
| -L | 锁定用户禁止其登陆系统 |
| -U | 解锁用户，允许其登陆系统 |
| -s | 变更默认终端 |
| -u | 修改用户的 UID |

**groupadd 命令用于创建群组，格式为：”groupadd [选项] 群组名”。**

创建名称为 linuxprobe 的用户群组：

[root@linuxprobe ~]# groupadd linuxprobe

## 2.9 打包压缩文件命令

**tar 命令用于对文件打包压缩或解压，格式为：“tar [选项] [文件]”。**

打包并压缩文件:“tar -czvf 压缩包名.tar.gz 文件名”

解压并展开压缩包:“tar -xzvf 压缩包名.tar.gz”

| 参数 | 作用 |
| --- | --- |
| -c | 创建压缩文件 |
| -x | 解开压缩文件 |
| -t | 查看压缩包内有那些文件 |
| -z | 用 Gzip 压缩或解压 |

| | |
|---|---|
| -j | 用 bzip2 压缩或解压 |
| -v | 显示压缩或解压的过程 |
| -f | 目标文件名 |
| -p | 保留原始的权限与属性 |
| -P | 使用绝对路径来压缩 |
| -C | 指定解压到的目录 |

**将/etc 目录内文件打包并通过 gzip 格式压缩：**

[root@linuxprobe ~]# tar czvf etc.tar.gz /etc
tar: Removing leading '/' from member names
/etc/
/etc/fstab
/etc/crypttab
/etc/mtab
/etc/fonts/
/etc/fonts/conf.d/
/etc/fonts/conf.d/65-0-madan.conf
/etc/fonts/conf.d/59-liberation-sans.conf
/etc/fonts/conf.d/90-ttf-arphic-uming-embolden.conf
/etc/fonts/conf.d/59-liberation-mono.conf
/etc/fonts/conf.d/66-sil-nuosu.conf
……………………………………

**将 etc.tar.gz 解压到/root/etc 目录中：**

[root@linuxprobe ost ~]# mkdir /root/etc
开始解压 etc.tar.gz 文件：
[root@linuxprobe ~]# tar xzvf etc.tar.gz -C /root/etc

### 2.10 文件查询搜索命令

**grep 命令用于对文本进行搜索，格式为："grep [选项] [文件]"。**

搜索某个关键词:"grep 关键词 文本文件"

| 参数 | 作用 |
|---|---|
| -b | 将可执行文件(binary)当作文本文件（text）来搜索 |
| -c | 仅显示找到的次数 |
| -i | 忽略大小写 |
| -n | 显示行号 |
| -v | 反向选择——仅列出没有"关键词"的行。 |

**搜索在/etc/passwd 中”/sbin/nologin”出现的行，找出系统中不允许登陆的用户。**

[root@linuxprobe ~]# grep /sbin/nologin /etc/passwd

bin:x:1:1:bin:/bin:/sbin/nologin

daemon:x:2:2:daemon:/sbin:/sbin/nologin

adm:x:3:4:adm:/var/adm:/sbin/nologin

lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin

mail:x:8:12:mail:/var/spool/mail:/sbin/nologin

operator:x:11:0:operator:/root:/sbin/nologin

………………………………

**找出文件“/usr/share/gedit/plugins/snippets/docbook.xml”中所有包含 entry 的行并输出到/root/lines：**

**答案模式:**grep entry /usr/share/gedit/plugins/snippets/docbook.xml >> /root/lines

**find 命令用于查找文件，格式为：“find [查找路径] 寻找条件 操作”。**

这里需要注意下 find 命令非常灵活导致参数非常复杂，这里不要求大家记住，用时来查即可。

对于常用搜索路径有几个小窍门：“~”代表用户的家目录，“.”代表当前目录，“/”代表根目录。

| 参数 | 作用 |
| --- | --- |
| -name | 匹配名称 |
| -perm | 匹配权限（mode 为完全匹配，-mode 为包含即可） |
| -user | 匹配所有者 |
| -group | 匹配所有组 |
| -mtime -n +n | 匹配修改内容的时间（-n 指 n 天以内，+n 指 n 天以前） |
| -atime -n +n | 匹配访问文件的时间-n 指 n 天以内，+n 指 n 天以前 |
| -ctime -n +n | 匹配修改权限的时间-n 指 n 天以内，+n 指 n 天以前 |
| -nouser | 匹配无所有者的文件 |
| -nogroup | 匹配无所有组的文件 |
| -newer f1 !f2 | 匹配比文件 f1 新却比 f2 旧的文件 |
| --type b/d/c/p/l/f | 匹配文件类型（块设备、目录、字符设备、管道、链接文件、文件文件） |
| --size | 匹配文件的大小（+50k 查找超过 50k 的文件,而-50k 则代表查找小于 50k 的文件） |
| -prune | 忽略某个目录 |
| --exec {} \; | 后面可接对搜索到结果进一步处理的命令（下面会有演示） |

**搜索在/etc/中所有以 host 开头的文件：**

其中的”host*”表示所有以 host 开头的文件：

[root@linuxprobe ~]# find /etc -name "host*" -print

/etc/avahi/hosts

/etc/host.conf

/etc/hosts

/etc/hosts.allow

/etc/hosts.deny

/etc/selinux/targeted/modules/active/modules/hostname.pp

/etc/hostname

**搜索整个系统中所有包含 SUID 的文件（因 SUID 的数字表示法是 4，而减号表示只要包含即可）。**

[root@linuxprobe ~]# find / -perm -4000 -print

/usr/bin/fusermount

/usr/bin/su

/usr/bin/umount

/usr/bin/passwd

/usr/sbin/userhelper

/usr/sbin/usernetctl

………………………………

**找出用户 linuxprobe 的文件并复制到/root/findresults 目录。**

重点是”-exec {} \;”其中的{}代表 find 命令搜索出的文件，记住结尾必须是\;

[root@linuxprobe ~]# find / -user linuxprobe -exec cp -arf {} /root/findresults/ \;

读者们，辛苦了~你们有没有有觉得 Linux 中的“命令”真的很方便？下章中将会正式的使用到它们，做好准备吧!另外大家也几乎见到了所有 Linux 系统中日常用到的命令，但这仅仅是打好基础，为了今后能更加高效的管理主机，请一定要学习后来的 Shell 脚本课程。

**本章结束，您可以在此写下笔记：**

# 第 3 章 管道符、重定向与环境变量。

**章节概述：**

Don’t be so excited!虽然此刻您已经学完了上百个常用Linux命令，但如前面所说：“光用命令本身并不能做好工作”。下个章节将学习 Shell 脚本的使用方法，所以本章节要有些承上启下的作用，理论知识点会比较多，但都很实用。
当读者学习完管道命令符、输入输出重定向、通配符以及环境变量后便可以将命令组合的更加恰当、高效率。

## 3.1 管道命令符

管道命令符“|”的作用是将前一个命令的标准输出当作后后一个命令的标准输入，格式为“命令 A|命令 B”。
例如前面章节学习的 **grep** 命令(文本搜索命令)，通过匹配关键词”**/sbin/nologin**“找出了所有被限制登陆系统的用户，如果我们希望统计所有不允许登陆系统的用户个数，该怎么做那(仅可用一条命令)?
首先理清思路:
找出被限制登陆用户的命令是:**grep “/sbin/nologin” /etc/passwd**
统计文本行数的命令则是:**wc -l**
现在要做的是就是将搜索命令的输出值传递给统计命令，其实只要把管道符夹在中间就可以了。
[root@linuxprobe ~]# grep "/sbin/nologin" /etc/passwd | wc -l
33
很厉害的功能吧!我们将原本将会显示到屏幕上的用户信息列表交给”**wc -l**“命令做了进一步的加工，简直太方便了!!
**用翻页的形式查看/etc 目录中有那些文件：**
[root@linuxprobe ~]# ls -l /etc/ | more
total 1400
………………省略部分文件………………
**向 linuxprobe 用户发送一封邮件：**
[root@linuxprobe ~]# echo "Content" | mail -s "Subject" linuxprobe
切换至 linuxprobe 用户：
[root@linuxprobe ~]# su - linuxprobex
Last login: Fri Jul 10 09:44:07 CST 2015 on :0
查看邮件箱，果然有一封邮件哦：
[linuxprobe@linuxprobe ~]$ mail
Heirloom Mail version 12.5 7/5/10. Type ? for help.
"/var/spool/mail/linuxprobe": 1 message 1 new
>N 1 root Sun Aug 30 17:33 18/578 "Subject"
**使用非交互式设置用户密码，将 root 的密码修改为 linuxprobe。**
[root@linuxprobe ~]# echo "linuxprobe" | passwd --stdin root
Changing password for user root.
passwd: all authentication tokens updated successfully.
当然读者们可不要误解管道命令符只能用一次哦，完全可以这样用：“命令 A|命令 B|命令 C”。

## 3.2 输入输出重定向

在学习标准输入输出重定向前，我们先看一个简单的演示吧：
查看 linuxprobe 目录的信息：
[root@linuxprobe ~]# ls linuxprobe/
查看 xxxxxx 目录的信息：
[root@linuxprobe ~]# ls xxxxxx/
ls: cannot access xxxxxx: No such file or directory
刚刚我们先查看了一个名为 linuxprobe 目录内的文件，后又尝试查看名为”xxxxxx”目录内的文件，显示该目录并不存在。

虽然好像命令都执行成功了，但其实有所差异，前者执行后返回的是标准输出，而后者执行失败返回的是错误输出。
标准输入(STDIN，文件描述符为 0)：默认从键盘输入，为 0 时表示是从其他文件或命令的输出。
标准输出(STDOUT，文件描述符为 1)：默认输出到屏幕，为 1 时表示是文件。
错误输出(STDERR，文件描述符为 2)：默认输出到屏幕，为 2 时表示是文件。

**对于输出重定向符有这些情况：**

| 符号 | 作用 |
| --- | --- |
| 命令 > 文件 | 将标准输出重定向到一个文件中（清空原有文件的数据） |
| 命令 2> 文件 | 将错误输出重定向到一个文件中（清空原有文件的数据） |
| 命令 >> 文件 | 将标准输出重定向到一个文件中（追加到原有内容的后面） |
| 命令 2>> 文件 | 将错误准输出重定向到一个文件中（追加到原有内容的后面） |
| 命令 >> 文件 2>$1 | 将标准输出与错误输出共同写入到文件中（追加到原有内容的后面） |

**对于输入重定向有这些情况：**

| 符号 | 作用 |
| --- | --- |
| 命令 < 文件 | 将文件作为命令的标准输入 |
| 命令 << 分界符 | 从标准输入中读入，直到遇见“分界符”才停止 |
| 命令 < 文件 1 > 文件 2 | 将文件 1 作为命令的标准输入并将标准输出到文件 2 |

**那么来做几个输出和输入重定向的实验吧：**

将 man 命令的帮助文档写入到/root/man.txt 中：

[root@linuxprobe ~]# man bash > /root/man.txt

向 readme.txt 文件中写入一行文字：

[root@linuxprobe ~]# echo "Welcome to LinuxProbe.Com" > readme.txt

向 readme.txt 中追加一行文字：

[root@linuxprobe ~]# echo "Quality linux learning materials" >> readme.txt

查看 readme.txt 中的内容：

[root@linuxprobe ~]# cat readme.txt
Welcome to LinuxProbe.Com
Quality linux learning materials

把 readme.txt 文件作为输入重定向给 wc -l 命令来计算行数，命令等同于“cat readme.txt | wc -l”。

[root@linuxprobe ~]# wc -l < readme.txt
2

用 echo、mail 和管道符命令(|)发给 linuxprobe 用户一封邮件，但内容只能有一句话。

[root@linuxprobe ~]# echo "Content" | mail -s "Subject" linuxprobe

如果您想像写信一样发邮件，就用输入重定向试试吧，向指定邮箱发送一封邮件，标题为 Readme，内容逐行输入。其中的 over 被称为分节符，是用户自定义的，当系统遇到这个分界符时会认为输入结束。

[root@linuxprobe ~]# mail -s "Readme" root@linuxprobe.com << over
> I think linux is very practical
> I hope to learn more
> can you teach me ?
> over

正常情况下输入分界符后会结束输入操作并发送邮件，不会有报错信息。

[root@linuxprobe ~]#

**这次咱们还是用”ls”命令查看文件信息，若文件不存在则将报错信息输出到/root/stderr.txt 中：**

[root@linuxprobe ~]# ls linuxprobe 2> /root/stderr.txt
-rw-r--r--. 1 root root 0 Mar 1 13:30 linuxprobe

文件为空，代表上面命令并没有报错：

[root@linuxprobe ~]# cat /root/stderr.txt

将查看 xxxxxx 目录命令的错误信息输出到/root/stderr.txt 文件中：

[root@linuxprobe ~]# ls xxxxxx 2> /root/stderr.txt

查看到 stderr.txt 文件中保存的 ls 命令报错信息：

[root@linuxprobe ~]# cat /root/stderr.txt
ls: cannot access xxxxxx: No such file or directory

因为”linuxprobe”的文件确实存在，所有没有报错信息，但”xxxxxx”文件是不存在的，所以则将报错信息输出到了指定的文件。

### 3.3 命令行通配符

例如我们想对一类文件批量操作，例如批量查看硬盘文件属性，那么正常命令会是：

[root@linuxprobe ~]# ls /dev/sda
[root@linuxprobe ~]# ls /dev/sda1
[root@linuxprobe ~]# ls /dev/sda2
[root@linuxprobe ~]# ls /dev/sda3

但有些时候确实不知道分区的个数或分区号，这时候就要用到通配符来搞定了，Bash 解释器的支持多种文本通配符包括：

| 通配符 | 含义 |
| --- | --- |
| * | 匹配零个或多个字符。 |
| ? | 匹配任意单个字符。 |
| [0-9] | 匹配范围内的数字。 |
| [abc] | 匹配已出的任意字符。 |

查看 sda 开头的所有设备文件：

[root@linuxprobe ~]# ls /dev/sda*
/dev/sda /dev/sda1 /dev/sda2

查看 sda 后面有一个字符的设备文件：

[root@linuxprobe ~]# ls /dev/sda?
/dev/sda1 /dev/sda2

查看 sda 后面包含 0-9 数字的设备文件：
[root@linuxprobe ~]# ls /dev/sda[0-9]
/dev/sda1 /dev/sda2
查看 sda 后面是 1 或 3 或 5 的设备文件：
[root@linuxprobe ~]# ls /dev/sda[135]
/dev/sda1
另外 bash 解释器还支持很多的特殊字符扩展：

| 字符 | 作用 |
| --- | --- |
| \(反斜杠) | 转义后面单个字符 |
| ''(单引号) | 转义所有的字符 |
| ""(双引号) | 变量依然生效 |
| ``(反引号) | 执行命令语句 |

定义名称为 PRICE 的变量值为 5：
[root@linuxprobe ~]# PRICE=5
想要输出"价格是 5"：
[root@linuxprobe ~]# echo "Price is $PRICE"
Price is 5
想要输出"价格是$5"，但因为美元符号与代表变量取值的$符号冲突了，所以报错了：
[root@linuxprobe ~]# echo "Price is $$PRICE"
Price is 3767PRICE
添加一个反斜杠，将第一个$符号转义：
[root@linuxprobe ~]# echo "Price is \$$PRICE"
Price is $5
使用单引号，变量将不再被取值：
[root@linuxprobe ~]# echo 'Price is \$$PRICE'
Price is \$$PRICE
执行 uname -a 后可以查看到本机内核的版本与架构信息（反引号里面的命令会被执行）：
[root@linuxprobe ~]# echo `uname -a`
Linux linuxprobe.com 3.10.0-123.el7.x86_64 #1 SMP Mon May 5 11:16:57 EDT 2014 x86_64 x86_64 x86_64 GNU/Linux

### 3.4 实用的 PATH 变量

**alias 命令用于设置命令的别名，格式为："alias 别名=命令"。**
例如担心复制文件时误将文件覆盖，那么执行 alias cp="cp -i"则每次覆盖都会询问用户。
**unalias 命令用于取消命令的别名，格式为："unalias 别名"。**
设置 cp 命令的别名：
[root@linuxprobe ~]# alias cp="cp -i"
取消 cp 命令的别名：
[root@linuxprobe ~]# unalias cp

如同前面所讲的——在 Linux 中所有的一切都是文件，**命令文件也不例外。**那当用户执行了一条”ls“命令后发生了什么事情?

步骤一:如果是以绝对/相对路径输入的命令则直接执行（如执行/bin/ls)。

步骤二:检查是否为 alias 别名命令。

步骤三:由 bash 判断其是“内部命令”还是“外部命令”。

内部命令：属于解释器内部的

外部命令：独立于解释器外的命令文件

步骤四：通过$PATH 变量中定义的路径进行命令查找。

查看$PATH 变量的方法:echo $PATH

/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin

如果您想知道某个命令是“内部命令”还是“外部命令”？执行执行“**type 命令名字**”，解释器就会告诉你呦~

$PATH 变量是“解释器的助手”，它负责告诉 bash 用户要执行的命令可能存放在那里，然后 bash 就会乖乖的在这些目录里寻找。

在变量$PATH 中目录之间用冒号“:”间隔开了，当然您也能自定义一些命令存放目录，比如/root/bin。

查看当前的$PATH 变量内容：

[root@linuxprobe ~]# echo $PATH

/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin

为变量增加新的值：

[root@linuxprobe ~]# PATH=$PATH:/root/bin

查看此时的$PATH 变量内容：

[root@linuxprobe ~]# echo $PATH

/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:/root/bin

谨慎而有经验的运维人员在接手一台 Linux 系统后一定会在执行命令前查看下$PATH 变量中是否有可疑的目录。

### 3.5 重要的环境变量

上面学到的**$PATH** 是不是很实用？在 Linux 系统中还有许多重要的环境变量，我们可以用 **env** 命令查看到它们。

变量是由固定的“**变量名**”与用户或系统设置的“**变量值**”两部分组成的，如果有需求可直接修改~

| 变量名称 | 作用 |
|---|---|
| HOME | 用户的主目录“家”。 |
| SHELL | 当前的 shell 是哪个程序 |
| HISTSIZE | 历史命令记录条数 |
| MAIL | 邮件信箱文件 |
| LANG | 语系数据 |
| RANDOM | 随机数字 |
| PS1 | bash 提示符 |
| HISTFILESIZE | history 命令存储数量 |
| PATH | 在路径中的目录查找执行文件 |

| EDITOR | 默认文本编辑器 |
| --- | --- |
| HOME | 用户主目录 |

**让我们通过变量来查看下当前用户的家目录是那个吧。**

因为当前是以 root 用户登陆，所以显示为/root：

[root@linuxprobe ~]# echo $HOME

/root

切换到 linuxprobe 用户：

[root@linuxprobe ~]# su - linuxprobe

Last login: Fri Feb 27 19:49:57 CST 2015 on pts/0

切换为 linuxprobe 用户后，同样的"$HOME"变量却显示出了不同的值：

[linuxprobe@linuxprobe ~]$ echo $HOME

/home/linuxprobe

假设需要设置一个变量"WORKDIR"，让每个用户执行"cd $WORKDIR"都登陆到/home/workdir 目录中，该如何做那？

定义方法：变量名称=新的值

查看方法：echo $变量名称

创建该目录：

[root@linuxprobe ~]# mkdir /home/workdir

如前面所介绍的方法设置变量：

[root@linuxprobe ~]# WORKDIR=/home/workdir

成功切换，好棒！

[root@linuxprobe ~]# cd $WORKDIR

[root@linuxprobe workdir]# pwd

/home/workdir

切换到 linuxprobe 用户：

[root@linuxprobe workdir]# su - linuxprobe

Last login: Fri Mar 20 20:52:10 CST 2015 on pts/0

好奇怪，为什么没有切换到/home/workdir 目录呢：

[linuxprobe@linuxprobe ~]$ cd $WORKDIR

用 echo 查看发现该变量为空值：

[linuxprobe@linuxprobe ~]$ echo $WORKDIR

现在的问题是**为什么某个用户设置的环境变量不能被其他用户使用**？原因就在于变量的作用范围。

**export 命令用于将局部变量提升为全局变量，格式为："export 变量名[=变量值]"。**

将 WORKDIR 变量设置为全局变量：

[root@linuxprobe ~]# export WORKDIR

切换为 linuxprobe 用户：

[root@linuxprobe workdir]# su linuxprobe

Last login: Fri Mar 20 21:52:10 CST 2015 on pts/0

很棒哦~成功的切换了目录：

[linuxprobe@linuxprobe ~]$ cd $WORKDIR

[linuxprobe@linuxprobe workdir]$pwd

/home/workdir

第 4 章 Vim 编辑器与 Shell 命令脚本。

**章节简述**:

本章节将教给您如何使用 Vim 编辑器来编写文档、配置主机名称、网卡参数以及 yum 仓库，熟练使用各个模式和命令快捷键。我们可以通过 Vim 编辑器将 Linux 命令放入合适的逻辑测试语句(if、for、while、case)后最终写出简单实用的 Shell 脚本。还可以通过使用 at 命令或配置 Crontab 计划任务服务让系统自动按时工作，让日常工作更加的高效自动化，一劳永逸哦~

## 4.1 了解 Vim 文本编辑器

**在 Linux 系统中配置应用服务，实际上就是在修改它的配置文件**（配置文件可能有多个，其中包含不同的参数），而且日常工作中也一定免不了编写文档的事情吧，这些都是要通过文本编辑器来完成的。

在热门 Linux 操作系统中都会默认安装一款超好用的文本编辑器——名字叫“**vim**”，**vim** 是 **vi** 编辑器的升级版。

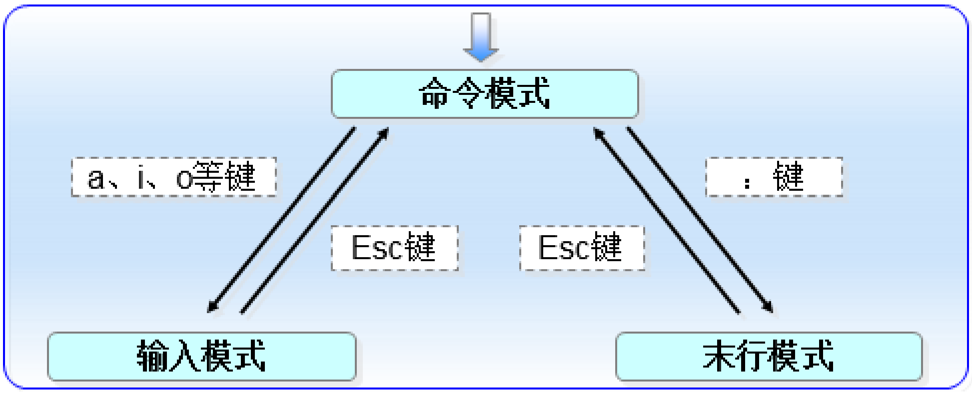
Vim 能够得到这么多厂商与用户的认可，原因就是在 Vim 编辑器中有三种模式——**命令模式**、**末行模式**和**编辑模式**，分别又有多种不同的命令快捷键组合，很大的提高了工作效率，用习惯后会觉得非常的顺手。要想在文本操作时更加高效率，我们必需先搞清 Vim 编辑器的三种模式的操作不同与切换方法。

命令模式：控制光标移动，可对文本进行删除、复制、粘贴等工作。

输入模式：正常的文本录入。

末行模式：保存、退出与设置编辑环境。

记住每次运行 vim 编辑器后都默认是“**命令模式**”，需要先进入到“**输入模式**”后再进行编写文档的工作，而每次编辑完成需先返回到“**命令模式**”后再进入“**末行模式**”对文本的保存或退出操作。




这里为大家总结出了最常用的快捷键命令，读者尽量记一下，忘记了来查也可以，至于“输入模式”则没有特殊技巧。

**vim 编辑器的命令模式中常用的快捷键**

| 命令 | 作用 |
|---|---|
| dd | 删除(剪切)光标所在整行。 |
| 5dd | 删除(剪切)从光标处开始的 5 行。 |
| yy | 复制光标所在整行。 |
| 5yy | 复制从光标处开始的 5 行。 |
| p | 将之前删除（dd）或复制（yy）过的数据粘贴到光标后。 |
| /字符串 | 在文本中从上至下搜索该字符串。 |
| ?字符串 | 在文本中从下至上搜索该字符串。 |

| n | 显示搜索命令定位到的下一个字符串。 |
| --- | --- |
| N | 显示搜索命令定位到的上一个字符串。 |
| u | 撤销上一步的操作 |

**vim 编辑器的末行模式中的常用命令**

| 命令 | 作用 |
| --- | --- |
| :w | 保存 |
| :q | 退出 |
| :q! | 强制退出（放弃对文本的修改内容） |
| :wq! | 强制保存退出 |
| :set nu | 显示行号 |
| :set nonu | 不显示行号 |
| :命令 | 执行该命令 |
| :整数 | 跳转到该行 |

**需要读者注意的两点：**

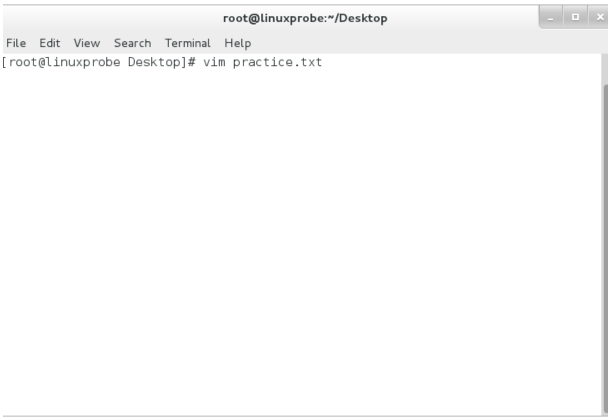
1.在命令模式与末行模式中，所有的快捷键参数均区分大小写。

2.在末行模式中所有快捷键参数前都有一个冒号”：“。

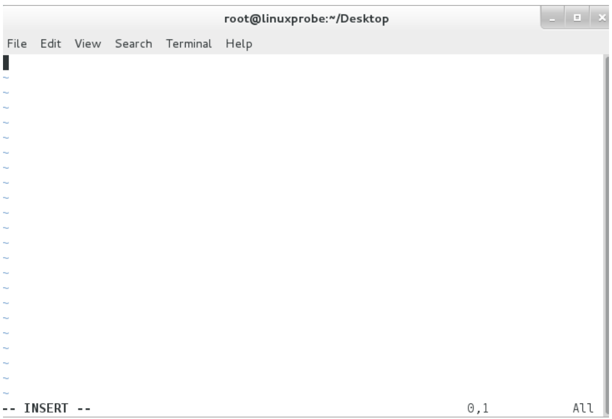
### 4.1.1 编写简单文档

现在动手编写一个文档吧、我会把每个步骤和按键尽量都标注出来，如果命令忘记了就回到上面小节再看看吧~**超简单!**

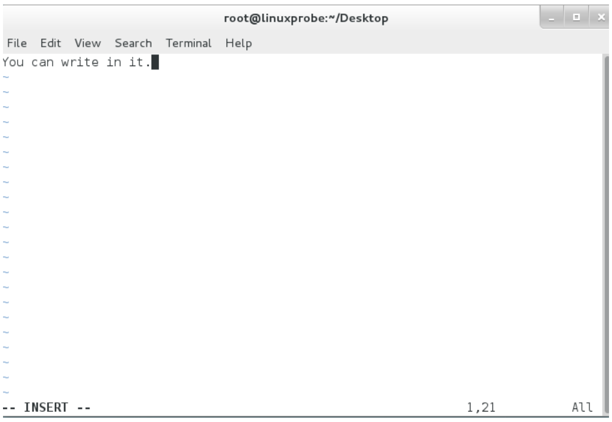
第 1 步：创建文档。



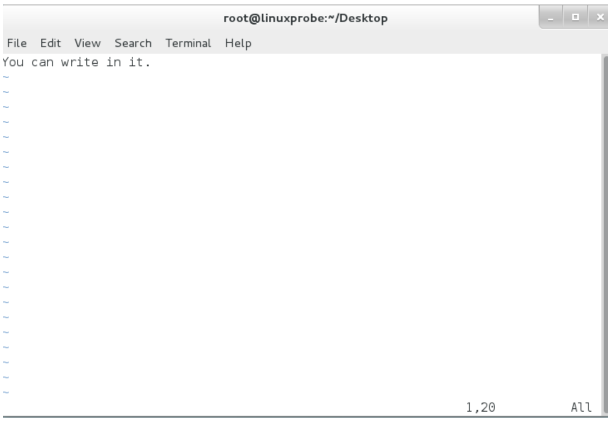

第 2 步：敲击字母“a”，进入输入模式。



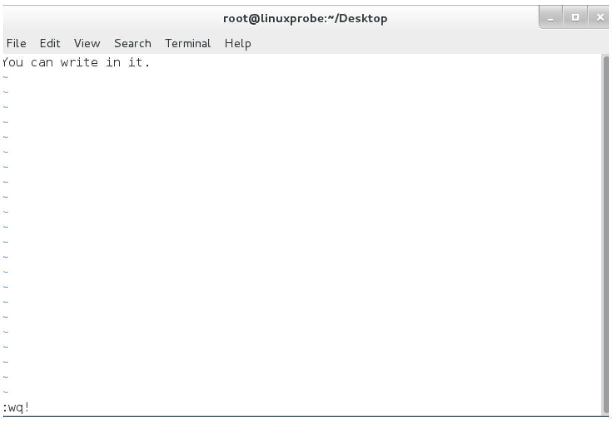
第 3 步：输入一行字符。



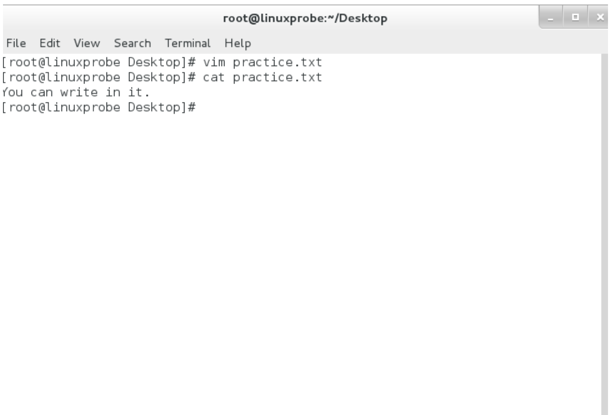

第 4 步：敲击[ESC]返回到命令模式。



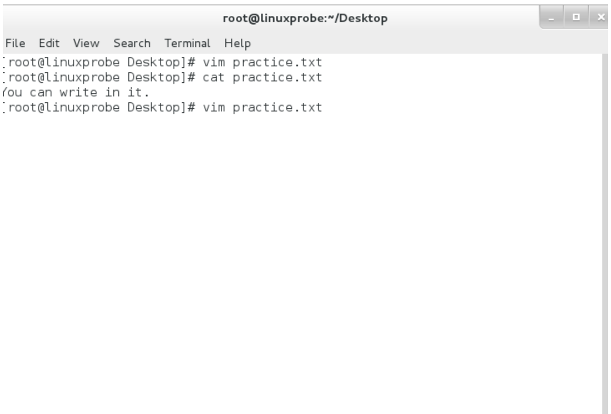

第 5 步：敲击“：wq”，保存并退出。



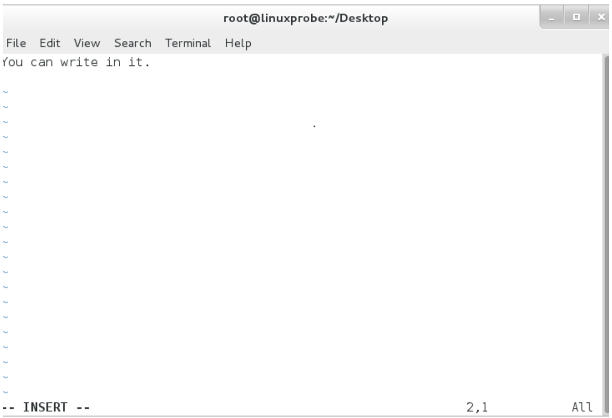

第 6 步：查看文档的内容。



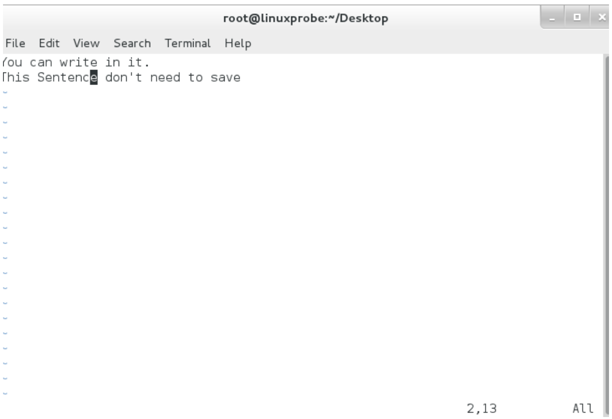

第 7 步：编辑文件。



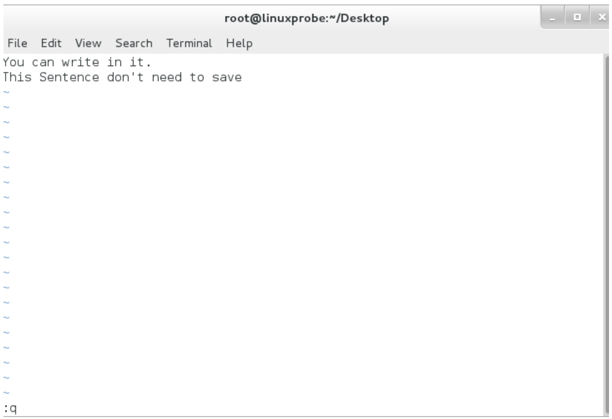

第 8 步：敲击字母“o”，进入到输入模式。



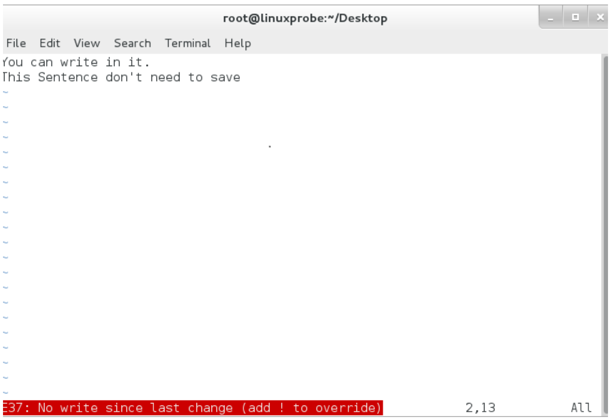
第 9 步：输入一行字符。



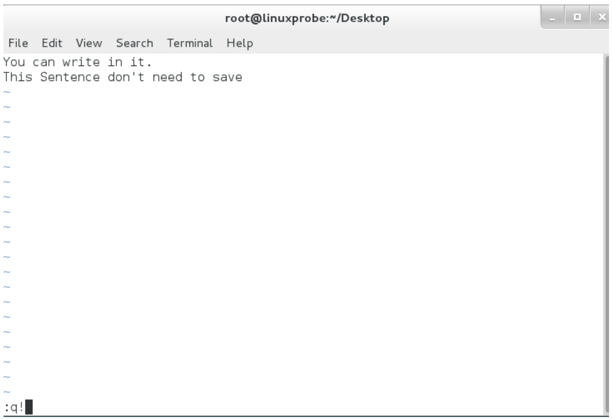

第 10 步：返回命令模式后尝试“：q”退出不保存。



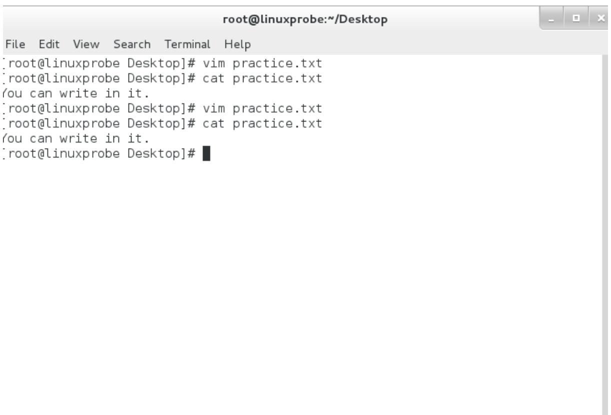

第 11 步：提示内容已被修改，不能退出。




第 12 步：敲击“：q!”后强制退出不保存。




第 13 步：查看文档，发现第二句确实没保存。




### 4.1.2 配置主机名称

红帽 RHEL7 系统的主机名称保存在/etc/hostname 文件中，我们要想将其修改为”linuxprobe.com“，思路大致如下：

第 1 步:使用 vim 命令修改”/etc/hostname“主机名称文件。

第 2 步:将原始主机名称删除后追加”linuxprobe.com“。

第 3 步:保存退出并用 hostname 命令检查是否修改成功。

使用 vim 命令编辑主机名称文件后末行模式执行:wq!后即可保存退出：

[root@linuxprobe ~]# vim /etc/hostname
linuxprobe.com
使用 hostname 命令查看当前的主机名称：
[root@linuxprobe ~]# hostname
linuxprobe.com

**4.1.3 配置网卡信息**

既然已经会用 vim 编辑器了，快来试试配置你的 Linux 系统网卡吧，不把网卡先配置妥当就不能与其他机器通信的。
在红帽 RHEL6 系统中网卡配置文件的前缀为”eth”，第 1 块即为”eth0″，第 2 块即为”eth1″并依此类推……
而在红帽 RHEL7 系统中网卡配置文件的前缀则为”ifcfg-eno“，例如”ifcfg-eno16777736″。
网卡的配置文件存放在**“/etc/sysconfig/network-scripts”**目录中。
在修改配置文件前，先来学些关键词术语吧：

**网卡类型:**TYPE=Ethernet
**地址分配模式:**BOOTPROTO=static
**网卡名称:**NAME=eno16777736
**是否启动:**ONBOOT=yes
**IP 地址:**IPADDR=192.168.10.10
**子网掩码:**NETMASK=255.255.255.0
**网关地址:**GATEWAY=192.168.10.1
**DNS 地址:**DNS1=192.168.10.1

上面的网卡配置文件代表着“这是一个以太网卡设备,名称为”eno16777736″且开机自动启动,IP 地址等信息需由人工指定”。
**配置网卡信息前先来理清思路:**

第 1 步:首先我们要切换到”/etc/sysconfig/network-scripts“目录中（该目录存放着网卡的配置文件)。
第 2 步:使用 vim 命令修改文件”**ifcfg-eno16777736**“。
第 3 步:逐项写入配置参数，并保存退出。
第 4 步:重新启动网卡命令:”**systemctl restart network**“。
第 5 步:通过 ping 命令测试网卡信息是否生效。

切换到网卡配置文件所在的目录：
[root@linuxprobe ~]# cd /etc/sysconfig/network-scripts/
编辑网卡配置文件并填入下面的信息：
[root@linuxprobe network-scripts]# vim ifcfg-eno16777736

```
TYPE=Ethernet
BOOTPROTO=static
NAME=eno16777736
ONBOOT=yes
IPADDR=192.168.10.10
NETMASK=255.255.255.0
GATEWAY=192.168.10.1
DNS1=192.168.10.1
```

重新启动网卡，正常情况不会提示信息:
[root@linuxprobe network-scripts]# systemctl restart network

不错哦，成功的 ping 通证明网卡配置正确并生效了：

```
[root@linuxprobe network-scripts]# ping 192.168.10.10
PING 192.168.10.10 (192.168.10.10) 56(84) bytes of data.
64 bytes from 192.168.10.10: icmp_seq=1 ttl=64 time=0.081 ms
64 bytes from 192.168.10.10: icmp_seq=2 ttl=64 time=0.083 ms
64 bytes from 192.168.10.10: icmp_seq=3 ttl=64 time=0.059 ms
64 bytes from 192.168.10.10: icmp_seq=4 ttl=64 time=0.097 ms
^C
--- 192.168.10.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.059/0.080/0.097/0.013 ms
```

#### 4.1.4 配置 Yum 仓库

既然对 vim 编辑器的理论已经学扎实，现在就来动手配置下 Yum 仓库吧~**先来理清思路**:

第 1 步:首先我们要切换到”/etc/yum.repos.d/“目录中（因为该目录存放着 yum 仓库的配置文件）

第 2 步:使用 vim 编辑器创建并打开一个名为 **rhel7.repo** 的新文件，名称可以自定义，但后缀必需为 **repo**。

第 3 步:逐项写入配置参数，并保存退出。

第 4 步:按配置参数的路径将光盘挂载。

第 5 步:将光盘挂载信息写入到/etc/fstab 文件中。

第 6 步:使用”**yum install httpd -y**“命令检查是否配置正确。

切换到/etc/yum.repos.d 目录中：

```
[root@linuxprobe ~]# cd /etc/yum.repos.d/
```

打开 Vim 界面后敲击”a”进入到插入模式：

编辑完成后敲击[ESC]并在末行模式中:wq!保存并退出。

```
[root@linuxprobe yum.repos.d]# vim rhel7.repo
 [rhel7]
 name=rhel7
 baseurl=file:///media/cdrom
 enabled=1
 gpgcheck=0
```

创建挂载光盘的目录：

```
[root@linuxprobe yum.repos.d]# mkdir -p /media/cdrom
```

仓库提供方式为本地，所以需要将光盘挂载到/media/cdrom 中：

```
[root@linuxprobe yum.repos.d]# mount /dev/cdrom /media/cdrom
mount: /dev/sr0 is write-protected, mounting read-only
```

设置成开机自动挂载：

```
[root@linuxprobe yum.repos.d]# vim /etc/fstab
/dev/cdrom /media/cdrom iso9660 defaults 0 0
```

测试安装”httpd”服务，出现“Complete”则代表 Yum 仓库配置正确：

```
[root@linuxprobe yum.repos.d]# yum install httpd
Loaded plugins: langpacks, product-id, subscription-manager
rhel7 | 4.1 kB 00:00
(1/2): rhel7/group_gz | 134 kB 00:00
(2/2): rhel7/primary_db | 3.4 MB 00:00
```

Resolving Dependencies
……………………………………………………
Complete!

## 4.2 了解 Shell 脚本

我曾经将 Shell 形容是人与计算机硬件的“翻译官”，Shell 作为用户与 Linux 系统通讯的媒介，自身也定义了各种变量与参数，并提供了诸如循环、分支等高级语言才有的控制结构特性。如何正确的使用这些功能，准确下达命令就显得尤为重要。

**Shell 的工作形式分为两种**

交互式(Interactive):用户输入一条命令，Shell 解释并执行一条。

批处理(Batch):用户事先编写一个 Shell 脚本(Script)，其中包含诸多命令，Shell 会一次执行完所有命令。

那么大家在前面学习 Linux 命令时，大致就是属于交换式了，**Shell 脚本是将各种命令通过逻辑语句组合而成的程序**。Shell 脚本需要用到很多的 Linux 命令以及结合之前学习过的正则表达法、管道命令以及数据流重定向等语法规则来完成指定任务。

**查看系统中所有可用的 Shell 解释器:**

```
[root@linuxprobe ~]# cat /etc/shells
    /bin/sh
    /bin/bash
    /sbin/nologin
    /usr/bin/sh
    /usr/bin/bash
    /usr/sbin/nologin
    /bin/tcsh
    /bin/csh
```

**查看当前的 Shell 解释器:**

```
[root@linuxprobe ~]# echo $SHELL
/bin/bash
```

### 4.2.1 编译简单的脚本

Shell 脚本的编写要使用到 Vim 文本编辑器，按照命令的执行顺序依次编写，每行写一条 Linux 命令。并且一个完整的 Shell 脚本则应该包括“**脚本声明**”、“**注释信息**”和“**可执行语句**”。

**脚本声明(#!)**:告知系统用何种 shell 来解释。

**注释信息(#)**:对可执行语句或程序功能做介绍，可以不写。

**可执行语句**:执行的具体命令。

先来编写一个简单的 Shell 脚本吧，功能是显示当前的工作路径并列出当前目录下的所有文件与属性。

```
[root@linuxprobe ~]# vim Example.sh
    #!/bin/bash
    #For Example BY linuxprobe.com
    pwd
    ls -al
```

原来编写 Shell 脚本如此的简单~执行脚本有三种方法：

**脚本文件路径**:./Example.sh

**sh 脚本文件路径**:sh Example.sh

**source 脚本文件路径**:source Example.sh

只要脚本文件路径没有写错，sh 或 source 命令都可以直接执行该脚本，但直接访问脚本路径的方式有点特殊。
使用直接访问脚本路径的方式提示出现错误，权限不足:

[root@linuxprobe ~]# ./Example.sh
bash: ./Example.sh: Permission denied

需要为脚本设置可执行权限后才能顺利运行：

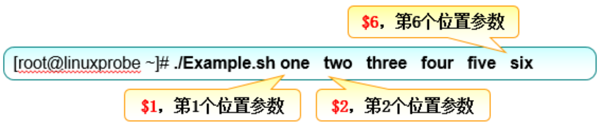
[root@linuxprobe ~]# chmod u+x Example.sh

再来运行就没有问题了：

[root@linuxprobe ~]# ./Example.sh
/root/Desktop
total 8
drwxr-xr-x. 2 root root 23 Jul 23 17:31 .
dr-xr-x---. 14 root root 4096 Jul 23 17:31 ..
-rwxr--r--. 1 root root 55 Jul 23 17:31 Example.sh

### 4.2.2 接收用户的参数

Shell 脚本为了能够让用户更灵活的完成工作需求，应该有办法接收用户输入的参数，像上面的脚本的写法真的很不灵活。
您在执行命令时的参数是不是像这样使用："命令名 **参数 1 参数 2 参数 3**"，所以其实在可执行文件中已经内设了接收用户参数的位置变量。




不光如此，还有这些已经被定义好的 Shell 预定义变量：

| | |
|---|---|
| $0 | 当前执行 Shell 脚本的程序名。 |
| $1-9,${10},${11}…… | 参数的位置变量。 |
| $# | 一共有多少个参数。 |
| $* | 所有位置变量的值。 |
| $? | 判断上一条命令是否执行成功，0 为成功，非 0 为失败。 |

好的~来动手完成一个可以接收用户参数的 Shell 脚本吧：

[root@linuxprobe ~]# vim Example.sh
#!/bin/bash
echo "当前脚本名称为$0"
echo "总共有$#个参数，分别是$*。"
echo "第 1 个参数为$1，第 5 个为$5。"

使用 sh 命令来执行脚本，并附带 6 个参数：

[root@linuxprobe ~]# sh Example.sh one two three four five six
当前脚本名称为 Example.sh
总共有 6 个参数，分别是 one two three four five six。
第 1 个参数为 one，第 5 个为 five。

### 4.2.3 判断用户的参数

Shell 脚本有时还要判断用户输入的参数，例如像 mkdir 命令一样，当目录不存在则创建，若已经存在则报错，条件测试语句能够测试特定的表达式是否成立，当条件成立时返回值为 0，否则返回其他数值。




**细分测试语句有:文件测试、逻辑测试、整数值比较、字符串比较。**

**文件测试:[ 操作符 文件或目录名]**

| 操作符 | 作用 |
| --- | --- |
| -d | 测试是否为目录。 |
| -e | 测试文件或目录是否存在。 |
| -f | 判断是否为文件。 |
| -r | 测试当前用户是否有权限读取。 |
| -w | 测试当前用户是否有权限写入。 |
| -x | 测试当前用户是否有权限执行。 |

测试/etc/fstab 是否为目录：

[root@linuxprobe ~]# [ -d /etc/fstab ]

显示上一条命令的返回值，非 0 则为失败，即不是目录：

[root@linuxprobe ~]# echo $?

1

测试/etc/fstab 是否为文件：

[root@linuxprobe ~]# [ -f /etc/fstab ]

显示上一条命令的返回值为 0，即 fstab 是文件：

[root@linuxprobe ~]# echo $?

0

符号&&代表逻辑上的”与“，当前面的命令执行成功才会执行后面的命令，判断/dev/cdrom 设备是否存在，若存在则输出 Exist：

[root@linuxprobe ~]# [ -e /dev/cdrom ] && echo "Exist"

Exist

**逻辑测试:[ 表达式 1 ] 操作符 [ 表达式 2 ]**

| 操作符 | 作用 |
| --- | --- |
| && | 逻辑的与，"而且"的意思。 |
| \|\| | 逻辑的或，"或者"的意思。 |
| ! | 逻辑的否。 |

USER 变量是当前登陆的用户名：

[root@linuxprobe ~]# echo $USER

root

若当前登陆的用户不是 root，则输出 user，执行后结果为空：

[root@linuxprobe ~]# [ $USER != root ] && echo "user"

登入用户 linuxprobe，再来测试便输出了 user 字样：

[root@linuxprobe ~]# su linuxprobe -

[linuxprobe@linuxprobe root]$ [ $USER != root ] && echo "user"

user

换回 root 用户后用加强版的判断语句，非 root 用户则输出 user，若是 root 则直接输出 root：

[root@linuxprobe ~]# [ $USER != root ] && echo "user" || echo "root"

root

这里请读者思考下&&与||的逻辑含义，因为前面的&&不成立，所有后面的||才会执行。

**整数值比较:[ 整数 1 操作符 整数 2 ]**

| 操作符 | 作用 |
|---|---|
| -eq | 判断是否等于 |
| -ne | 判断是否不等于 |
| -gt | 判断是否大于 |
| -lt | 判断是否小于 |
| -le | 判断是否等于或小于 |
| -ge | 判断是否大于或等于 |

比较 10 是否大于 10：

[root@linuxprobe ~]# [ 10 -gt 10 ]

显示上一条命令执行失败，10 不大于 10：

[root@linuxprobe ~]# echo $?

1

比较 10 是否等于 10：

[root@linuxprobe ~]# [ 10 -eq 10 ]

显示上一条命令执行成功，10 等于 10：

[root@linuxprobe ~]# echo $?

0

获取当前可用的内存量，并将此值赋值给变量 FreeMem，逐个解释下吧~

首先用 free -m 查看以 m 为单位的内存使用情况，然后 grep cache:过滤出剩余内存的行，最后用 awk '{print $3}' 过滤只保留第三列，而 FreeMem='语句'则表示执行里面的语句后赋值给变量。

[root@linuxprobe ~]# FreeMem='free -m | grep cache: | awk '{print $3}''

验证变量是否已经获得可用内存量：

[root@linuxprobe ~]# echo $FreeMem

609

判断此值是否小与 1024(单位是 M)，若小于则提示内存不足：

[root@linuxprobe ~]# [ $FreeMem -lt 1024 ] && echo "Insufficient Memory"

Insufficient Memory

**字符串比较:[字符串 1 操作符 字符串 2]**

| 操作符 | 作用 |
|---|---|
| = | 比较字符串内容是否相同。 |
| != | 比较字符串内容是否不同。 |
| -z | 判断字符串内容是否为空。 |

判断 String 变量是否为空值：

[root@linuxprobe ~]# [ -z $String ]

上一条命令执行成功，说明变量 String 确实为空值：

[root@linuxprobe ~]# echo $?

0

输出当前的系统语言:

[root@linuxprobe ~]# echo $LANG

en_US.UTF-8

判断当前的系统语言是否为英文，否则输出“不是英语”：

[root@linuxprobe ~]# [ $LANG != "en.US" ] && echo "Not en.US"

Not en.US

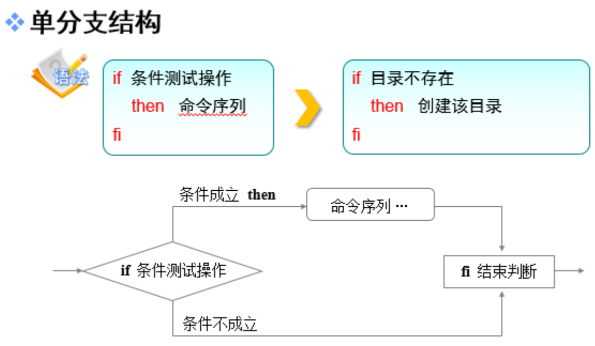
## 4.3 条件测试语句

条件测试语句能够让 Shell 脚本根据实际工作灵活调整工作内容，例如判断系统的状态后执行指定的工作，或创建指定数量的用户，批量修改用户密码，这些都可以让 Shell 脚本通过条件测试语句完成。

**if 条件语句**

if 条件语句分为单分支结构、双分支结构、多分支结构，复杂度逐级上升，但却可以让 Shell 脚本更加的灵活。

首先来说单分支结构，仅用 **if**、**then**、**fi** 关键词组成，只在条件成立后执行。



**单分支 if 语句:**判断目录是否存在，若不存在则自动创建。

编写 Shell 脚本并写入下面的语句：

```
[root@linuxprobe ~]# vim Example.sh
  #!/bin/bash
  DIR="/media/cdrom"
  if [ ! -e $DIR ]
  then
  mkdir -p $DIR
  fi
```

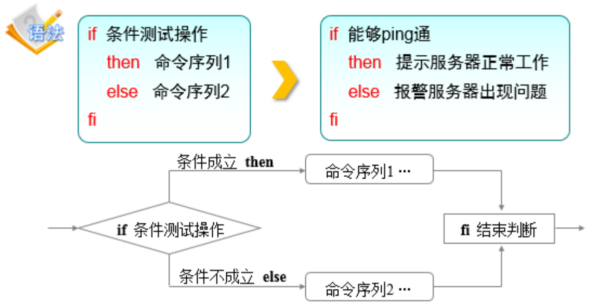
执行后默认没有回显，读者可动手添加 echo 语句显示创建过程：

```
[root@linuxprobe ~]# sh Example.sh
```

查看该目录是否被创建：

```
[root@linuxprobe ~]# ls -d /media/cdrom
/media/cdrom
```

双分支结构是由 **if**、**then**、**else**、**fi** 关键词组成，做条件成立或条件不成立的判断。




**双分支 if 语句:**判断指定主机能否 ping 通，根据返回结果分别给予提示或警告。

为了减少用户的等待时间，需要为 ping 命令追加-c 参数代表发送数据包的个数，-i 代表每 0.2 秒发一个数据包，-W 则为 3 秒即超时。而$1 为用户输入的第一个参数（IP 地址），$?为上一条命令的执行结果，判断是否等于 0(即成功)。

```
[root@linuxprobe ~]# vim Example.sh
  #!/bin/bash
  ping -c 3 -i 0.2 -W 3 $1 &> /dev/null
  if [ $? -eq 0 ]
  then
  echo "Host $1 is up."
  else
  echo "Host $1 is down."
  fi
```

给予脚本可执行权限，否则请用 sh 或 source 命令执行：

```
[root@linuxprobe ~]# chmod u+x Example.sh
```

参数为要检测的主机 IP 地址，根据返回值判断为 up：

```
[root@linuxprobe ~]# ./Example.sh 192.168.10.10
Host 192.168.10.10 is up.
```

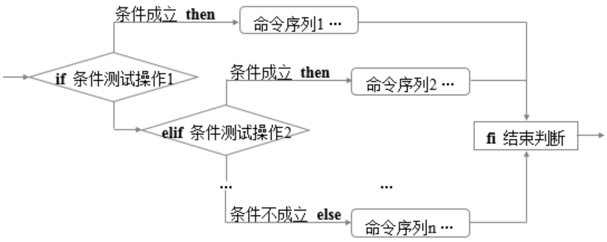
根据 ping 命令的执行结果判断主机出现网络故障。

```
[root@linuxprobe ~]# ./Example.sh 192.168.10.20
```

Host 192.168.10.20 is down.

多分支结构相对就比较复杂了，是由 if、then、else、elif、fi 关键词组成，根据多种条件成立的可能性执行不同的操作。




**多分支 if 语句**:判断用户输入的分数在那个区间内，然后判定为优秀、合格或不合格。

**read 命令用于将用户的输入参数赋值给指定变量，格式为：”read -p [提示语句] 变量名”。**

使用 read 命令让用户为 GRADE 变量赋值，判断分数必需同时满足大于 85 且小于 100 才输出 Excellent，判断分数必需同时满足大于 70 且小于 84 才输出 Pass，其余所有的情况均会输出 Fail。

[root@linuxprobe ~]# vim Example.sh

```
#!/bin/bash
read -p "Enter your score（0-100）: " GRADE
if [ $GRADE -ge 85 ] && [ $GRADE -le 100 ] ; then
echo "$GRADE is Excellent"
elif [ $GRADE -ge 70 ] && [ $GRADE -le 84 ] ; then
echo "$GRADE is Pass"
else echo "$GRADE is Fail"
fi
```

给予脚本可执行权限，否则请用 sh 或 source 命令执行：

[root@linuxprobe ~]# chmod u+x Example.sh

输入 88 分，满足第一判断语句，所以输出 Excellent：

[root@linuxprobe ~]# ./Example.sh

Enter your score（0-100）：88

88 is Excellent

输入 80 分，满足第二判断语句，所以输出 Pass：

[root@linuxprobe ~]# ./Example.sh Enter your score（0-100）：80

80 is Pass

输入 30 与 200 分都属于其他情况，所以输出 Fail：

[root@linuxprobe ~]# ./Example.sh  Enter your score（0-100）：30

30 is Fail

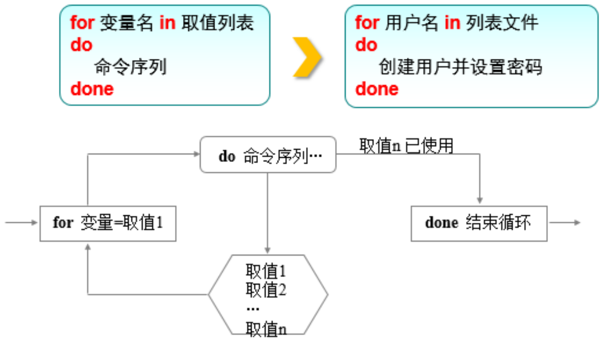
[root@linuxprobe ~]# ./Example.sh Enter your score（0-100）：200

200 is Fail

请您动手在上面 Shell 脚本中添加判断语句，将所有小于 0 分或大于 100 分的输入都予以警告。

## for 条件语句

for 条件语句会先读取多个不同的变量值，然后逐一执行同一组命令。




**for 条件语句**:从列表文件中读取用户名，逐个创建用户并将密码设置。

创建用户名称列表文件：

```
[root@linuxprobe ~]# vim users.txt
andy
barry
carl
duke
eric
george
```

Shell 脚本提示用户输入要设置的密码并赋值给 PASSWD 变量，从 users.txt 文件中读入用户名并赋值给 UNAME 变量，而查看用户的信息都重定向到/dev/null 文件，不显示到屏幕。

```
[root@linuxprobe ~]# vim Example.sh
#!/bin/bash
read -p "Enter The Users Password : " PASSWD
for UNAME in ‘cat users.txt‘
do
id $UNAME &> /dev/null
if [ $? -eq 0 ]
then
echo "Already exists"
else
useradd $UNAME &> /dev/null
echo "$PASSWD" | passwd --stdin $UNAME &> /dev/null
if [ $? -eq 0 ]
then
echo "Create success"
else
echo "Create failure"
fi
fi
done
```

执行批量创建用户的 Shell 脚本程序，输入为用户设定的密码口令，检查脚本是否为我们完成创建用户的动作：

```
[root@linuxprobe ~]# source Example.sh
Enter The Users Password : linuxprobe
Create success
Create success
Create success
Create success
Create success
Create success
[root@linuxprobe ~]# tail -6 /etc/passwd
andy:x:1001:1001::/home/andy:/bin/bash
barry:x:1002:1002::/home/barry:/bin/bash
carl:x:1003:1003::/home/carl:/bin/bash
duke:x:1004:1004::/home/duke:/bin/bash
eric:x:1005:1005::/home/eric:/bin/bash
george:x:1006:1006::/home/george:/bin/bash
```

这个 Shell 脚本还存在一个小小的遗憾，它只会输出帐号创建成功或失败，但没有指明是那个帐号，这个功能请读者动手添加下，记得是要用$UNAME 变量哦。

**for 条件语句**:从列表文件中读取主机地址，逐个测试是否在线。

首先创建主机地址列表：

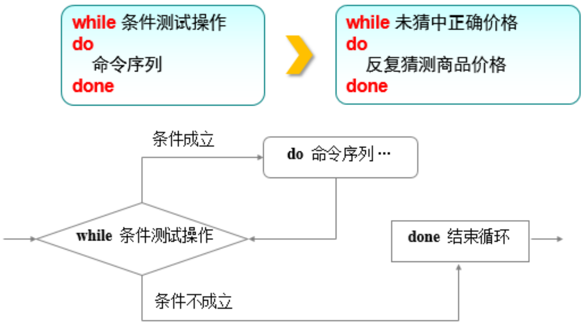
```
[root@localhost ~]# vim ipadds.txt
    192.168.10.10
    192.168.10.11
    192.168.10.12
```

这个脚本可以参考前面双分支 if 语句———从 ipadds.txt 中读取主机地址后赋值给 HLIST 变量后逐个 ping 列表中的主机 IP 地址测试主机是否在线：

```
[root@localhost ~]# vim Example.sh
    #!/bin/bash
    HLIST=$(cat ~/ipadds.txt)
    for IP in $HLIST
    do
    ping -c 3 -i 0.2 -W 3 $IP &> /dev/null
    if [ $? -eq 0 ] ; then
    echo "Host $IP is up."
    else
    echo "Host $IP is down."
    fi
    done
    [root@linuxprobe ~]# ./Example.sh
    Host 192.168.10.10 is up.
    Host 192.168.10.11 is down.
    Host 192.168.10.12 is down.
```

**while 条件语句**

while 条件语句用于重复测试某个条件，当条件成立时则继续重复执行。




**while 条件语句**:随机生成一个 0-999 的整数，判断并提示用户输入的值过高或过低，只有当用户猜中才结束程序。脚本中的$RANDOM 是一个随机变量，用于在%1000 后会得到一个介于 0-999 的整数后赋值给 PRICE 变量，while 后面的 true 代表该循环会永久循环执行：

```
#!/bin/bash
PRICE=$(expr $RANDOM % 1000)
TIMES=0
echo "商品实际价格为 0-999 之间，猜猜看是多少？"
while true
do
read -p "请输入你猜测的价格数目：" INT
let TIMES++
if [ $INT -eq $PRICE ] ; then
echo "恭喜你答对了，实际价格是 $PRICE"
echo "你总共猜测了 $TIMES 次"
exit 0
elif [ $INT -gt $PRICE ] ; then
echo "太高了！"
else
echo "太低了！"
fi
done
```

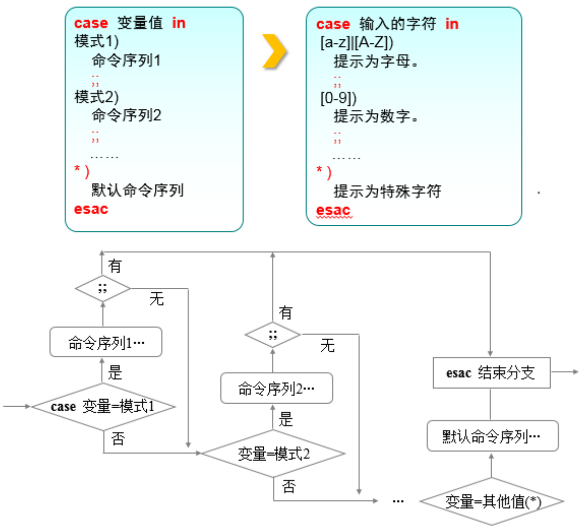
动手试试运行 Shell 脚本吧，每次 RANDOM 变量的值都是随机的：

```
[root@linuxprobe ~]# chmod u+x Example.sh
[root@linuxprobe ~]# ./Example.sh
    商品实际价格为 0-999 之间，猜猜看是多少？
    请输入你猜测的价格数目：500
    太低了！
    请输入你猜测的价格数目：800
    太高了！
    请输入你猜测的价格数目：650
    太低了！
```

请输入你猜测的价格数目：720
太高了！
请输入你猜测的价格数目：690
太低了！
请输入你猜测的价格数目：700
太高了！
请输入你猜测的价格数目：695
太高了！
请输入你猜测的价格数目：692
太高了！
请输入你猜测的价格数目：691
恭喜你答对了，实际价格是 691
你总共猜测了 9 次

**case 条件语句**

case 条件语句可以依据变量的不同取值，分别执行不同的命令动作。




**case 条件语句**:提示用户输入一个字符，判断该字符是字母、数字或特殊字母。

提示用户输入一个字符并将其赋值给变量 KEY，判断变量 KEY 为何种字符后分别输出是字母、数字还是其他字符：

[root@linuxprobe ~]# vim Example.sh

```
#!/bin/bash
read -p "请输入一个字符，并按 Enter 键确认：" KEY
case "$KEY" in
[a-z]|[A-Z])
echo "您输入的是 字母。"
;;
[0-9])
echo "您输入的是 数字。"
;;
*)
echo "您输入的是 空格、功能键或其他控制字符。"
```

esac

[root@linuxprobe ~]# chmod u+x Example.sh
[root@linuxprobe ~]# ./Example.sh
请输入一个字符，并按 Enter 键确认：6
您输入的是 数字。
[root@linuxprobe ~]# ./Example.sh
请输入一个字符，并按 Enter 键确认：p
您输入的是 字母。
[root@linuxprobe ~]# ./Example.sh
请输入一个字符，并按 Enter 键确认：^[[15~
您输入的是 空格、功能键或其他控制字符。

**4.4 计划任务服务**

有经验的系统运维工程师能够让系统自动化运行，无需人工的干预就可以让各个服务、命令在指定的时间段运行、停止。实际上这些操作都是由系统的计划任务功能完成的，而计划任务又分为“一次性”与“长期性”之分，可以理解为：
一次性计划任务:今晚 11 点 30 分开启网站服务（例如新网站的公测）
长期性计划任务:每周 1、3、5 的凌晨 3 点 25 分将/home/wwwroot 目录打包备份为 backup.tar.gz
先来讲一次性任务吧，它是由 atd 服务/进程来实现的，计划的管理操作是”at”命令，具体的可用参数如下：

| 参数 | 作用 |
| --- | --- |
| at <时间> | 安排一次性任务 |
| atq 或 at -l | 查看任务列表 |
| at -c 序号 | 预览任务与设置环境 |
| atrm 序号 | 删除任务 |

一般用 at 命令创建计划任务有交互式与非交互式两种方法，先来看看交换式的方法，（输完成后敲击 Ctrl+d 来保存退出）：
[root@linuxprobe ~]# at 23:30
at > systemctl start httpd
at >
job 3 at Mon Apr 27 23:30:00 2015
[root@linuxprobe ~]# atq
3 Mon Apr 27 23:30:00 2015 a root
直接用 echo 语句将要执行的命令传送给 at 命令:
[root@linuxprobe ~]# echo "systemctl start httpd" | at 23:30
job 4 at Mon Apr 27 23:30:00 2015
[root@linuxprobe ~]# atq
3 Mon Apr 27 23:30:00 2015 a root
4 Mon Apr 27 23:30:00 2015 a root
删除的时候只需要用 atrm 命令与任务编号就可以啦~
[root@linuxprobe ~]# atrm 3

[root@linuxprobe ~]# atrm 4

[root@linuxprobe ~]# atq

对于创建长期可循环的计划任务，则要用到 cron 服务啦，具体使用方法如下：

创建、编辑计划任务:crontab -e [-u 用户名]

查看计划任务:crontab -l [-u 用户名]

删除计划任务:crontab -r [-u 用户名]

其中在创建、编辑计划任务时有个固定的格式，请读者们一定要记住。




| 字段 | 说明 |
| --- | --- |
| 分钟 | 取值为从 0 到 59 之间的整数 |
| 小时 | 取值为从 0 到 23 之间的任意整数 |
| 日期 | 取值为 1 到 31 之间的任意整数 |
| 月份 | 取值为 1 到 12 之间的任意整数 |
| 星期 | 取值为 0 到 7 之间的任意整数，其中 0 与 7 均为星期日 |
| 命令 | 要执行的命令或程序脚本 |

需要用 cron 计划任务实现的功能:”每周 1、3、5 的凌晨 3 点 25 分将/home/wwwroot 目录打包备份为 backup.tar.gz”

编辑 root 用户自己的计划任务：

[root@linuxprobe ~]# crontab -e

no crontab for root - using an empty one

crontab: installing new crontab

使用”crontab -l”命令查看计划任务的内容：

[root@linuxprobe ~]# crontab -l

25 3 * * 1,3,5 /usr/bin/tar -czvf backup.tar.gz /home/wwwroot

如果想对某个用户设置多个计划任务，则可直接用”**crontab -e**“命令将命令逐条添加即可，让计划任务自动在每周 1-5 的凌晨 1 点打包网站目录后自动清除/tmp 目录下的所有文件::

[root@linuxprobe ~]# crontab -e

crontab: installing new crontab

[root@linuxprobe ~]# crontab -l

25 3 * * 1,3,5 /usr/bin/tar -czvf backup.tar.gz /home/wwwroot

0 1 * * 1-5 /usr/bin/rm -rf /tmp/*

**本章结束，您可以在此写下笔记：**

## 第 5 章 用户身份与文件权限。

**章节简述：**

详细的为读者讲述了用户、用户组和其余人在系统中的不同身份与能力，以及文件的读(r)写(w)执行(x)权限的作用。为了让系统更加的安全还需要学习 SUID、SGID 和 SBIT 的文件特殊权限，文件隐藏权限以及 ACL 访问控制列表。学会 su 命令和 sudo 服务后一定能够满足您以非超级用户操作实验或日常工作的需求，同时也保证了系统的安全性。

### 5.1 用户身份与能力

类 Unix 系统的设计初衷就是为让多用户同时工作，所以也迫使 Linux 系统有了极强的安全性，在前面安装红帽 RHEL7 操作系统时还特别要求“设置 root 用户密码”，而 root 用户是存在于所有类 UNIX 系统中的”超级用户“。

root 用户拥有极高的系统所有权，能够管理系统的各项功能，如**添加/删除用户**，**启动/关闭进程**，**开启/禁用硬件设备**等权限。虽然使用 root 用户工作时不会受到权限的控制，但老话讲“能力越大，责任就越大”，一旦我们使用这个高能的 root 用户敲出错误的命令就有可能毁掉整个系统，真得好好权衡下啊。

而其实”root“只是个名字，真正让它成为“超级用户”的是 UID 值：

**UID（即 User IDentification 的缩写）：每个用户都有对应的 UID 值，就像我们的身份证号码。**

超级用户 UID0:root 用户默认为 0。

系统用户 UID1-999:系统中系统服务由不同用户运行，更加安全，默认被限制登陆系统。

普通用户 UID1000~:即管理员创建的用于日常工作而不能管理系统的普通用户。

注意 UID 一定是不能冲突的，管理员创建的普通用户 UID 从 1000 开始（即便前面有闲置的号码）

帐户名称与 UID 保存在/etc/passwd 文件中，而帐户密码则保存在/etc/shadow 文件中。

**GID(即 Group IDentification 的缩写)：可将多个用户加入某个组中，方便指派任务或工作。**

想象公司员工如果想要在同部门内共享资料，就可以加入自己的工作组如技术部、运维部、财务部……

每个用户在被创建时均会创建一个默认组（其 GID 与 UID 相同，俗称基本组）而后加入的则叫扩展组，一定要分清楚。

用户组名称与 GID 保存在/etc/group 文件中。

### 5.2 文件权限与归属

Linux 系统中一切都是文件，文件和目录的所属与权限——**来分别规定所有者、所有组、其余人的读，写，执行权限。**

读(read)，写(write)，执行 e（xecute）简写即为(r,w,x)，亦可用数字(4,2,1)表示

| 权限项 | 读 | 写 | 执行 | 读 | 写 | 执行 | 读 | 写 | 执行 |
|---|---|---|---|---|---|---|---|---|---|
| 字符表示 | r | w | x | r | w | x | r | w | x |
| 数字表示 | 4 | 2 | 1 | 4 | 2 | 1 | 4 | 2 | 1 |
| 权限分配 | 文件所有者 | | | 文件所属组 | | | 其他用户 | | |

举例:如果某文件权限为 7 则代表可读，可写，可执行(4+2+1)。若权限为 6(4+2)则代表可读，可写。

那么权限为 5 与 3 时分别代表了什么？**想出答案后用鼠标选中下行即出答案（答案模式）**

**答案：**权限为 5 代表可读(4)和可执行(1)。而权限为 3 代表可写(2)和可执行(1)。

例如下图中的文件所有者(属主)为 root,所有组(属组)为 root，文件名为 instsall.log，权限位的第一个减号”－“代表的是文件类型：

**-:普通文件，d:目录文件，l:链接文件，b:块设备文件，c:字符设备文件，p:管道文件**



文件的权限为 rw-r－r－也就是分别表示**所有者(属主)有读写权限，所有组(属组)有读权限，其余人也仅有读权限。**
这个时候发现问题了吗？对于目录文件的读和写权限我们还可以理解，**目录要能执行操作?**
普通文件即实际保存数据的地方，其并不具备删除自身的权限：

r:可读取文件的实际内容

w:可编辑/新增/修改该文件的实际内容

x:可被执行

目录文件即保存有目录结构和文件权限：

r:可读取目录结构和权限

w:可更改目录结构列表、新建/删除/重命名/转移子文件/目录。

x:表示用户可进入到该目录中

### 5.3 文件的特殊权限

单纯对文件位置的 rwx 权限肯定不能满足我们对安全、便捷工作的需求，所以便有了 SUID 与 SGID 的特殊权限机制。
**SUID:让执行者临时拥有属主的权限（仅对拥有执行权限的二进制程序有效）**
比如所有用户都可以执行用于修改用户密码的 **passwd** 命令，但用户密码保存在**/etc/shadow** 文件中，默认权限是 **000** 即除了超级用户 root 外的所有用户都没有查看或编辑该文件的权限，所以对 **passwd** 命令加上 **SUID** 权限位，则可让普通用户临时获得程序所有者的身份，即以 root 用户的身份将变更的密码信息写入到 **shadow** 文件中。
SGID:
**功能一：让执行者临时拥有属组的权限（对拥有执行权限的二进制程序设置）**
举例来说**/dev/kmem** 是一个字符设备文件，用于存储内核程序要访问的数据，权限为：
**cr－r**—－　1 root system 2，1 Feb 11 2015　kmem
读者们看出问题了吗？除了以 **root 身份**或 **system 组成员**的用户都没有读取该文件的权限，但用户又需要使用系统的 ps 命令来查看系统进程状态，所以为了让用户能够获取到系统状态信息，ps 命令的权限被加了 **SGID** 位：
-r-xr-**s**r-x　1 bin system 59346 Feb 11 2015　ps
这样因为被给予了 SGID 权限，所以当用户执行了 ps 命令，实际有效用户组就是 **system** 啦，于是便能够顺利的读取设备文件啦~
**功能二：在该目录中创建的文件自动继承此目录的用户组（只可以对目录设置）**
比如我们将某个部门的工作目录给予了SGID权限，这样所有人创建的文件都归相同的工作组，这样方便以后的管理。
**chmod 命令用于修改文件或目录的权限，格式为：”chmod [参数] 权限 文件或目录名称”。**
**chown 命令用于修改文件或目录的所属主与所属组，格式为：“chown [参数] 所属主:所属组 文件或目录名称”。**
chmod 与 chown 的命令参数很简单记——对于文件不加参数，遇到目录加大写-**R**(递归，修改目录内所有文件的属性)。
创建工作目录并给予 GID 权限：

```
[root@linuxprobe ~]# cd /tmp
[root@linuxprobe tmp]# mkdir testdir
[root@linuxprobe tmp]# ls -ald testdir/
drwxr-xr-x. 2 root root 6 Feb 11 11:50 testdir/
[root@linuxprobe tmp]# chmod -Rf 777 testdir/
[root@linuxprobe tmp]# chmod -Rf g+s testdir/
[root@linuxprobe tmp]# ls -ald testdir/
drwxrwsrwx. 2 root root 6 Feb 11 11:50 testdir/
```

切换至普通用户 linuxprobe，在该目录创建文件：

```
[root@linuxprobe tmp]# su - linuxprobe
Last login: Wed Feb 11 11:49:16 CST 2015 on pts/0
```

[linuxprobe@linuxprobe ~]$ cd /tmp/testdir/
[linuxprobe@linuxprobe testdir]$ echo "linuxprobe.com" > test
[linuxprobe@linuxprobe testdir]$ ls -al
total 8
drwxrwsrwx. 2 root root 17 Feb 11 11:50 .
drwxrwxrwt. 18 root root 4096 Feb 11 11:50 ..
-rw-rw-r--. 1 linuxprobe root 15 Feb 11 11:50 test

**SBIT(Sticky Bit):只可管理自己的数据而不能删除他人文件(仅对目录有效)**

一般老师希望学生可以将作业上传到某个特定目录——但为了避免某些小破坏份子，想限制删除其他人文件的话，**那就要**设置 SBIT 位了，当然也可以叫做特殊权限位之**粘滞位**。

切换至普通用户，进入 tmp 目录：

[root@linuxprobe tmp]# su - linuxprobe
Last login: Wed Feb 11 12:41:20 CST 2015 on pts/0
[linuxprobe@linuxprobe ~]$ cd /tmp

查看目录权限，最后的 t 就是代表的粘滞位：

[linuxprobe@linuxprobe tmp]$ ls -ald
drwxrwxrwt. 17 root root 4096 Feb 11 13:03 .

创建一个文件吧：

[linuxprobe@linuxprobe tmp]$ echo "for test " > test
[linuxprobe@linuxprobe tmp]$ chmod -Rf 777 test

给予这个文件最大的权限(rwxrwxrwx,777)：

[linuxprobe@linuxprobe tmp]$ ls -al test
-rwxrwxrwx. 1 linuxprobe linuxprobe 10 Feb 11 12:59 test

此时切换到普通用户 blackshield 尝试删除该文件：

[root@linuxprobe tmp]# su - blackshield
Last login: Wed Feb 11 12:41:29 CST 2015 on pts/1
[blackshield@linuxprobe ~]$ cd /tmp
[blackshield@linuxprobe tmp]$ rm test
rm: remove write-protected regular file ‘test’? y
rm: cannot remove ‘test’: Operation not permitted

删除该文件时会提示错误，所以即便权限很充足，但因为特殊权限 SBIT 的缘故所以依然无法删除其他人的文件。

### 5.4 文件的隐藏属性

文件权限除了读写执行与 SUID、SGID、SBIT 外还有一种隐藏权限，例如明明有权限删除某个文件却报错了，或者仅能为某个文件追加内容而不能减少内容，遇到这种很“奇怪”的文件，就要怀疑是文件被设置隐藏权限了。

**chattr 命令用于设置文件的隐藏权限，格式为：“chattr [参数] 文件”。**

| 参数 | 作用 |
| --- | --- |
| i | 将无法对文件进行修改,若对目录设置后则仅能修改子文件而不能新建或删除。 |
| a | 仅允许补充（追加）内容.无法覆盖/删除(Append Only)。 |
| S | 文件内容变更后立即同步到硬盘(sync)。 |

| | |
|---|---|
| s | 彻底从硬盘中删除，不可恢复(用 0 填充原文件所在硬盘区域)。 |
| A | 不再修改这个文件的最后访问时间(atime)。 |
| b | 不再修改文件或目录的存取时间。 |
| D | 检查压缩文件中的错误。 |
| d | 当使用 dump 命令备份时忽略本文件/目录。 |
| c | 默认将文件或目录进行压缩。 |
| u | 当删除此文件后依然保留其在硬盘中的数据，方便日后恢复。 |
| t | 让文件系统支持尾部合并（tail-merging)。 |
| X | 可以直接访问压缩文件的内容。 |

**lsattr 命令用于显示文件的隐藏权限，格式为："lsattr [参数] 文件"。**

| 参数 | 作用 |
|---|---|
| a | 显示所有文件和目录。 |
| l | 显示隐藏属性的全称（默认简写成一个字母)。 |
| R | 递归处理，将指定目录下的所有文件及子目录一并处理。 |
| d | 若目标文件为目录，请加此参数。 |

写入一个名为 linuxprobe,内容为"for Test"的普通文件：

[root@linuxprobe ~]# echo "for Test" > linuxprobe

尝试用 rm 命令删除，结果成功：

[root@linuxprobe ~]# rm linuxprobe

rm: remove regular file ‘linuxprobe’? y

再次写入 linuxprobe 文件：

[root@linuxprobe ~]# echo "for Test" > linuxprobe

添加仅允许追加的隐藏权限（无法删除与覆盖）：

[root@linuxprobe ~]# chattr +a linuxprobe

再来尝试删除发现已经报错：

[root@linuxprobe ~]# rm linuxprobe

rm: remove regular file ‘linuxprobe’? y

rm: cannot remove ‘linuxprobe’: Operation not permitted

而用 ls 也无法看到不同的地方：

[root@linuxprobe ~]# ls -al linuxprobe

-rw-r--r--. 1 root root 9 Feb 12 11:42 linuxprobe

用 lsattr 命令则原形毕露了，果然是因为这个隐藏权限：

[root@linuxprobe ~]# lsattr linuxprobe

-----a---------- linuxprobe

去除对 linuxprobe 文件设置的隐藏权限：
[root@localhost ~]# chattr -a linuxprobe
再来看下 linuxprobe 文件的隐藏权限（已经没有了）：
[root@localhost ~]# lsattr linuxprobe
---------------- linuxprobe
尝试删除该文件（已经可以顺利删除了）：
[root@localhost ~]# rm linuxprobe
rm: remove regular file ‘linuxprobe’? **y**

### 5.5 su 命令与 sudo 服务

我建议读者使用非超级用户的普通身份来操作实验或日常工作，这样会更加的安全，当执行了错误的命令后也很少会让系统完全崩溃，但因为许多的系统管理命令只有超级用户才可以使用，所以无疑也让用户受到更多的束缚，如何让普通用户执行这些程序呢？

**su 命令用于变更使用者的身份(切换登陆者)，格式为："su [-] 用户名"。**

root 用户切换到其他用户时无需输入密码，尝试切换到普通用户 linuxprobe：
[root@linuxprobe ~]# su linuxprobe
成功切换后查看环境变量：
[linuxprobe@linuxprobe root]$ echo $PATH
/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:/root/bin
普通用户再切换用户需要输入对方帐户密码才可以：
[linuxprobe@linuxprobe root]$ su root
Password:
若需将环境变量改变为新用户的，请加参数"-"：
[root@linuxprobe ~]# su - linuxprobe
Last login: Mon Aug 24 19:27:09 CST 2015 on pts/0
再次查看环境变量：
[linuxprobe@linuxprobe ~]$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/linuxprobe/.local/bin:/home/linuxprobe/bin
从 su 切换用户模式中退出：
[linuxprobe@linuxprobe ~]$ exit
logout
回到 root 用户的身份：
[root@linuxprobe ~]#

**sudo 命令用于给普通用户提供额外权利来完成原本超级用户才能完成的任务，格式为："sudo [参数] 命令名称"。**

上面 su 命令允许普通用户完全变更为超级管理员的身份，但这也无疑会让系统增添很多的安全隐患，我们使用 sudo 程序可以仅将特定的命令/程序执行权限赋予给指定的用户，同时也避免了过多使用 root 身份，只要合理的配置 sudo 功能便可以合理的兼顾系统的安全性和用户便捷性，给读者的原则：
在保证普通用户完成工作的前提下，尽可能少的给予额外的权限。

**总结来说 sudo 的特色功能有：**
1:限制用户执行指定的命令。
2:记录用户执行的每一条命令。
3:配置文件（/etc/sudoers）提供集中的管理用户、权限与主机等参数。

4:验证过密码后 5 分钟(**默认值**)内无须再让用户验证密码，更加的方便。

**sudo 命令的常用参数包括有：**

| 参数 | 作用 |
|---|---|
| -h | 列出帮助信息。 |
| -l | 列出当前用户可执行的命令。 |
| -u 用户名或 UID 值 | 以指定的用户身份执行命令。 |
| -k | 清空安全时间，下次执行 sudo 时需要再次密码验证。 |
| -b | 在后台执行指定的命令。 |
| -p | 更改询问密码的提示语。 |

只用超级用户才可以使用 visudo 命令编辑 sudo 程序的配置文件（**/etc/sudoers**），visudo 命令的优势：
防止多个用户同时修改 sudo 配置文件。
对 sudo 程序配置文件的语法检查。
visudo 会调用 vi 编辑器来修改配置文件，而如果语法有报错则会报错：

```
visudo: >>> /etc/sudoers: syntax error near line 111 <<<
What now?
Options are:
(e)dit sudoers file again
(x)it without saving changes to sudoers file
(Q)uit and save changes to sudoers file (DANGER!)
```

此时可以敲击”**e 键**“来修正内容，敲击”**x 键**“直接退出不保存还可敲击”**Q 键**“强制保存退出（sudo 程序将不能被启动）。

**实验环节——允许 linuxprobe 用户执行所有命令：**

使用 visudo 命令编辑 sudo 程序的配置文件，在第 99 行添加参数允许 linuxprobe 用户能够从任意主机执行任意命令的参数。

格式为:允许使用 sudo 服务的主机　以谁的身份执行命令　具体可执行命令的列表

[root@linuxprobe ~]# visudo
linuxprobe ALL=(ALL) ALL

将上面的配置文件保存退出后切换至 linuxprobe 用户：

[root@linuxprobe ~]# su - linuxprobe
Last login: Thu Sep 3 15:12:57 CST 2015 on pts/1

查看 linuxprobe 用户可以使用那些 sudo 执行的命令（此处验证执行用户的密码）：

[linuxprobe@linuxprobe ~]$ sudo -l
[sudo] password for linuxprobe:
Matching Defaults entries for linuxprobe on this host:
requiretty, !visiblepw, always_set_home, env_reset, env_keep="COLORS
DISPLAY HOSTNAME HISTSIZE INPUTRC KDEDIR LS_COLORS", env_keep+="MAIL PS1
LC_IDENTIFICATION LC_MEASUREMENT LC_MESSAGES", env_keep+="LC_MONETARY
LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE", env_keep+="LC_TIME LC_ALL

LANGUAGE LINGUAS _XKB_CHARSET XAUTHORITY",
secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin
告知 linuxprobe 用户能够执行的命令有“所有”：
User linuxprobe may run the following commands on this host:
(ALL) ALL
使用 ls 命令查看/root 目录内的文件提示权限拒绝：
[linuxprobe@linuxprobe ~]$ ls /root
ls: cannot open directory /root: Permission denied
使用 sudo 命令以 root 用户身份执行则正常浏览：
[linuxprobe@linuxprobe ~]$ sudo ls /root
anaconda-ks.cfg Documents initial-setup-ks.cfg Pictures Templates
Desktop Downloads Music Public Videos

**实验环节——仅允许 linuxprobe 用户以 root 用户身份执行 cat 命令。**

使用 visudo 命令编辑 sudo 程序的配置文件，将前面实验的参数删除(第 99 行)。然后在第 112 行追加允许 linuxprobe 用户只能以 root 用户身份执行 cat 命令的参数：
[root@linuxprobe ~]# visudo
linuxprobe ALL=(root) /bin/cat
切换至 linuxprobe 用户:
[root@linuxprobe ~]# su - linuxprobe
Last login: Thu Sep 3 15:51:01 CST 2015 on pts/1
使用 cat 命令查看密码文件后提示权限不足：
[linuxprobe@linuxprobe ~]$ cat /etc/shadow
cat: /etc/shadow: Permission denied
使用 sudo 命令来运行 cat 命令后获得了 root 权限后查看成功：
[linuxprobe@linuxprobe ~]$ sudo cat /etc/shadow
root:$6$GV3UVtX4ZGg6ygA6$J9pBuPGUSgZslj83jyoI7ThJla9ZAULku3BcncAYF00Uwk6Sqc4E36MnD1hLtlG9Q
linuxprobe:$6$IaqSJH8ES4KGQp.7$NojzuWzxwKvgfufCN5CmYTaaMdiYYWDZwgoV0qgx6/K2ZSQUjby3lmkMvD
LuLIqbkuGsnVp1w.Z7S2kvWjHY6/:16626:0:99999:7:::

**实验环节——允许 linuxprobe 用户以任意身份执行命令，且每次都不需要密码验证。**

使用 visudo 命令编辑 sudo 程序的配置文件，将前面实验的参数删除（第 112 行）后在此行追加下面的参数
linuxprobe ALL=NOPASSWD: ALL
切换至 linuxprobe 用户：
[root@linuxprobe ~]# su - linuxprobe
清空安全时间：
Last login: Thu Sep 3 15:58:31 CST 2015 on pts/1
[linuxprobe@linuxprobe ~]$ sudo -k
执行 sudo 后不再需要密码验证：
[linuxprobe@linuxprobe ~]$ sudo ifconfig
eno16777728: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.10.10 netmask 255.255.255.0 broadcast 192.168.10.255
inet6 fe80::20c:29ff:fe9c:6373 prefixlen 64 scopeid 0x20<link>
ether 00:0c:29:9c:63:73 txqueuelen 1000 (Ethernet)
RX packets 264 bytes 40883 (39.9 KiB)
RX errors 0 dropped 0 overruns 0 frame 0

TX packets 31 bytes 4381 (4.2 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 0 (Local Loopback)
RX packets 2 bytes 140 (140.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 2 bytes 140 (140.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

### 5.6 文件访问控制列表

不知读者有没有发现其实上面讲解的 rwx 权限、特殊权限、隐藏权限都是对**某一类用户设置的**，而如果希望对**某个指定的用户**进行单独的权限设置，那么就需要用文件的访问控制列表来实现啦。

我们可以基于普通文件或目录设置进行设置 ACL，**通俗来说 ACL 就是设置指定的特定用户或用户组对某个文件的操作权限**。并且如果对某个目录设置了访问控制策略，那么子文件则继承其访问策略，而若对文件设置了访问控制策略则不再继承上级目录的控制策略。

**setfacl 命令用于增加或者修改 ACL 规则，格式为："setfacl [参数] 文件"。**

| 参数 | 作用 |
| --- | --- |
| -R | 递归(对目录使用) |
| -m | 设置文件的 acl 规则 |
| -b | 删除 acl 规则 |

**getfacl 命令用于显示文件的 ACL 规则，格式为："getfacl 文件"。**

[root@linuxprobe ~]# getfacl /root

**linuxprobe 用户因工作的原因需要有能读取 root 家目录文件的权限：**

[root@linuxprobe ~]# su - linuxprobe
Last login: Sat Mar 21 16:31:19 CST 2015 on pts/0

切换到 linuxprobe 用户：

[linuxprobe@linuxprobe ~]$ cd /root
-bash: cd: /root: Permission denied

尝试进入 root 用户的家目录失败了（当然进不去啦)：

[linuxprobe@linuxprobe root]$ exit

返回到 root 用户后设置 linuxprobe 对/root 有 rwx 权限：

[root@linuxprobe ~]# setfacl -Rm u:linuxprobe:rwx /root

切换到 linuxprobe 用户：

[root@linuxprobe ~]# su - linuxprobe
Last login: Sat Mar 21 15:45:03 CST 2015 on pts/1

成功进入到/root 目录：

[linuxprobe@linuxprobe ~]$ cd /root
[linuxprobe@linuxprobe root]$ ls

anaconda-ks.cfg Downloads Pictures Public

读者们也来试试看，能不能看到该文件的内容吧：

[linuxprobe@linuxprobe root]$ cat anaconda-ks.cfg

返回到 root 用户：

[linuxprobe@linuxprobe root]$ exit

用 getfacl 看到确实多了一条 user:linuxprobe:rwx：

[linuxprobe@linuxprobe ~]# getfacl /root

```
# file: .
# owner: root
# group: root
user::r-x
user:linuxprobe:rwx
group::r-x
mask::rwx
other::---
```

**本章结束，您可以在此写下笔记：**

第 6 章 存储结构与磁盘划分。

**章节简述：**

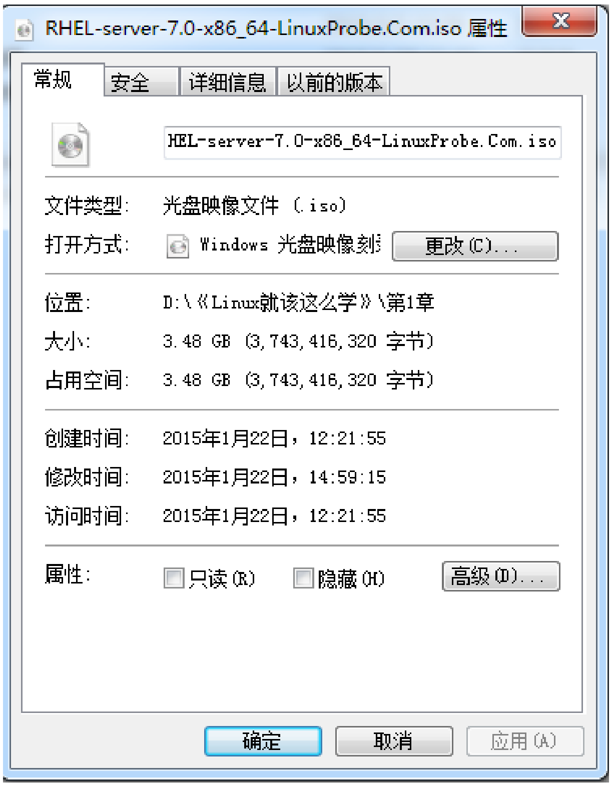
本章节从 Linux 系统的存储结构引入，讲述硬盘存储结构、硬件命名规则以及内核 Udev 设备管理器服务。

让读者理解文件系统的作用，能够区分 ext3,ext4,xfs 有何不同并学习将硬盘设备分区、格式化以及挂载等常用硬盘管理操作。

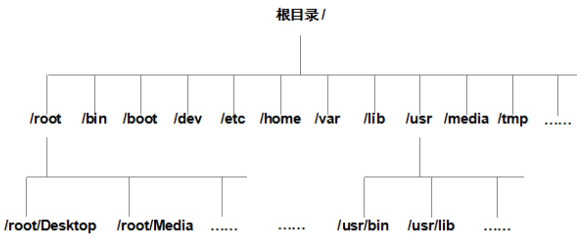
完整配置 SWAP 交换分区、quota 服务限制磁盘配额、ln 命令创建软/硬链接、RAID 磁盘阵列(0、1、5 和 10)、LVM 逻辑卷管理器。

## 6.1 一切从“/”开始

这是一张在 Windows™操作系统中文件的属性图，你能准确的找到它吗？




要想找到这个镜像文件则需要依次进入“D 盘”再进入“《Linux 就该这么学》”目录中的“第一章”目录，但在类 Unix 系统中并不存在 C/D/E/F 盘符呦，一切的文件都是从“根(/)”目录开始的并按照文件系统目录标准 FHS 采用树形结构来存放文件并定义了每个区域的用途。




目录名称严格的区分大小写，例如 root、rOOt、Root、rooT 等等均代表是不同的独立目录，并且名称中不得包含反斜杠(/)。

**主要常见的目录定义：**

| 目录名称 | 应放置文件的内容 |
| --- | --- |
| /boot | 开机所需文件——内核,开机菜单及所需配置文件等 |
| /dev | 任何设备与接口都以文件形式存放在此目录 |
| /etc | 配置文件 |
| /home | 用户主目录 |
| /bin | 单用户维护模式下还能够被操作的命令 |
| /lib | 开机时用到的函数库及/bin 与/sbin 下面命令要调用的函数 |
| /sbin | 开机过程中需要的 |
| /media | 一般挂载或删除的设备 |
| /opt | 放置第三方的软件 |
| /root | 系统管理员的主文件夹 |
| /srv | 一些网络服务的数据目录 |
| /tmp | 任何人均可使用的“共享”临时目录 |
| /proc | 虚拟文件系统，例如系统内核，进程，外部设备及网络状态等 |
| /usr/local | 用户自行安装的软件 |
| /usr/sbin | 非系统开机时需要的软件/命令/脚本 |
| /usr/share | 帮助与说明文件，也可放置共享文件。 |
| /var | 主要存放经常变化的文件，如日志。 |
| /lost+found | 当文件系统发生错误时，将一些丢失的文件片段存放在这里 |

另外一个重要的概念“路径”，这个路径指的是如何找到某个文件，分为“**绝对路径**”与“**相对路径**”：

**绝对路径(absolute)**:由根目录(/)开始写起的目录或文件名

**相对路径(relative)**:相对于当前路径的写法

**举例来说一个美国人想找下厕所，你有两种回答的方法。**

**绝对路径：**首先坐飞机来到中国，到了北京出首都机场做地铁到十号线潘家园站，出站坐 34 路到农光里下车路口左转。

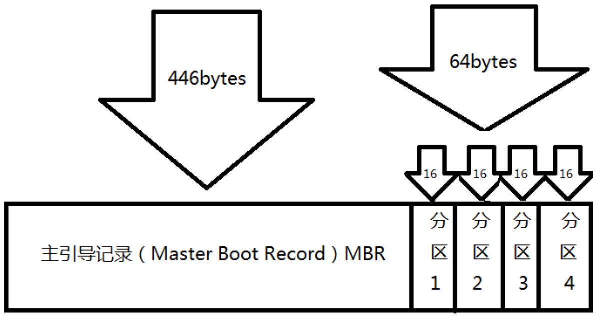
**相对路径：**前面路口左转

如果你说的是绝对路径，那么任何一个外国人都可以按照这个提示找到厕所，**但缺点是过于繁琐**，如果说的是相对

路径，那么这个美国人并不是在每个路口左转都能找到厕所，**缺点是不具备普遍性**。

### 6.2 物理设备的命名规则

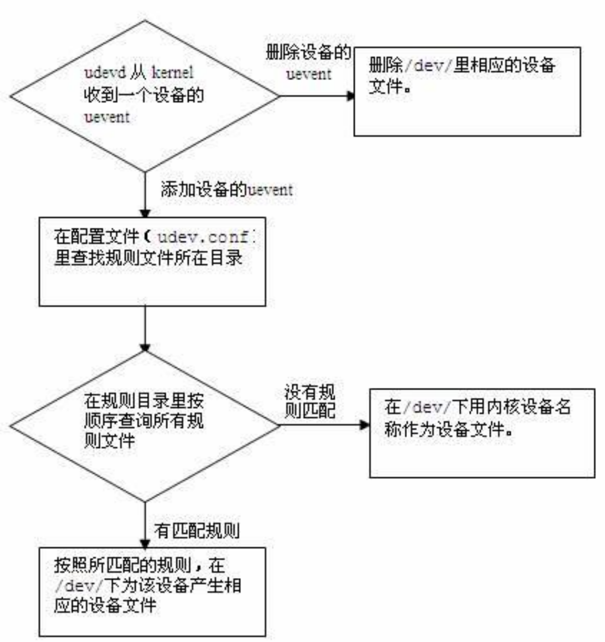
因为计算机中有了硬盘设备才使得我们游戏通关过后可以保存记录而不是再重新开始，硬盘设备则是由大量的“**扇区**”组成的，其中第一个扇区保存着**主引导记录与分区表信息**。单个扇区容量为 512bytes 组成，主引导记录需要占用 446bytes，分区表的为 64bytes，而每记录一个分区信息需要 16bytes，那么问题来了，**好像只能记录 4 个分区信息？**




所以运维人员一般会选择用 3 个主分区加 1 个扩展分区的方法，扩展分区中能够创建无限个逻辑分区，这样我们就可以用逻辑分区来满足多分区的需求了，**当然这里大家只需明白为什么主分区不能超过 4 个。**

Linux 系统中一切都是文件，那么硬件也不外乎。既然是文件就必须有名称啦，系统内核的设备管理器(Udev)会自动将硬件名称规范起来，让我们可以通过设备名称猜出设备大致的属性以及分区信息等，Udev 会一直以守护进程的形式运行并侦听来自内核发出的 uevent 来管理/dev 目录下的设备文件。

Udev 会根据内核发出的 uevent 来动态添加或删除/dev 目录中的设备文件，命名流程如下：

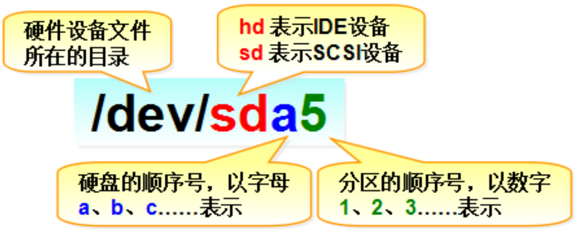



**常见的硬件命名如下：**

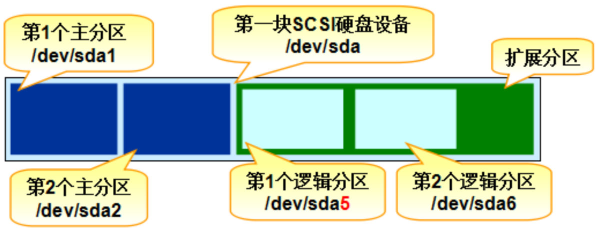
| 硬件设备 | 文件名称 |
|---|---|
| IDE 设备 | /dev/hd[a-d] |
| SCSI/SATA/U 盘 | /dev/sd[a-p] |

| 软驱 | /dev/fd[0-1] |
|---|---|
| 打印机 | /dev/lp[0-15] |
| 光驱 | /dev/cdrom |
| 鼠标 | /dev/mouse |
| 磁带机 | /dev/st0 或/dev/ht0(IDE 设备) |

因为现在的 IDE 设备已经很少见啦，所以一般硬盘设备都会是以“/dev/sd”开头的，而一台主机上可以有多块硬盘，系统便会用 a-p 来代表 16 块不同的硬盘（默认从 a 开始分配）且分区编号也很有讲究。
主分区编号从 1 开始至 4 结束，按顺序（也可指定分配数字）。
逻辑分区从编号 5 开始按顺序（也可指定分配数字）。
**那么来分析下”/dev/sda5“代表着什么硬件设备吧~**




1.首先/dev 目录下的都是硬件。
2.其次 sd 开头的是存储设备。
3.然后 a 代表第一个被识别到的设备。
4.最后 5 代表它是逻辑分区。
简单来讲:“**这是第一块硬盘设备中编号为 5 的逻辑分区**”，物理中的存储会是这样的：




6.3 文件系统与数据资料

文件管理系统的作用是将硬盘合理的规划，使得用户能够在上面正常建立文件、写入，读取，修改，转存文件与控制文件，而在 Linux 系统中支持超过数十种文件管理系统可供选择，常见的如下：

Ext3 是一款日志文件系统能够在异常停机中避免文件系统资料不一致的情况，自动修复数据的不一致与错误，然而一般重整文件系统相当耗费时间（尤其容量大的硬盘），当然也不能保证 100%资料不流失。它将会将整个磁盘的写入动作预先记录下来（每个细节），所以在异常停机后可以回溯追踪到被中断的部分。

Ext4 可以成为 Ext3 的后继版本，作为 RHEL6 系统的默认文件管理系统，其支持更大的文件系统到 1EB（1EB=1,073,741,824GB 且能够有无限多的子目录），另外 Ext4 文件系统能够批量分配 block 块并作”Extents”极大的提高了读写效率。

XFS 作为 RHEL7 默认的文件管理系统，它的日志型文件管理系统的优势在意外关机后尤其明显，可以快速的恢复可能被破坏的文件，另外经过优化后日志功能对硬盘性能影响非常小，同时最大支持 18EB 的存储容量满足了几乎所有需求。

讲课的时候我喜欢举得一个例子，希望能够帮助大家理解这个概念。

**“当我们拿到了一张大白纸，首先为了使用方便要裁剪，然后为了书写工整要先画格。”**
这里的“白纸”就是原始的硬盘而“裁剪”意味着分区，然后的“画格”就是格式化啦，最后写入内容。
因为硬盘要保存的数据实在太多了，所以一定要有个叫 super block 的“**硬盘地图**”并在上面记录着整个文件系统的信息，但绝不可能把数据直接写到这个大地图中，因为这样的话会导致它“很大”，查询与写入速度会变得非常慢，于是每个文件的权限与属性都会记录在 inode table 中（每个文件都会占用一个独立的 inode 表格，默认为 128bytes)，记录着：

该文件的访问权限(read,write,execute)
该文件的所属主与组(owner,group)
该文件的大小(size)
该文件的创建或状态修改时间(ctime)
该文件的最后一次访问时间(atime)
该文件的修改时间(mtime)
文件的特殊权限(SUID,SGID,SBIT)
**该文件的真实数据地址(point)**

而实际的数据则保存在 block 块中（大小可以是 1K、2K 或 4K)，下面的说明中，我们以 4K 为例。
情况一：文件体积很小（1K)，那么依然会占用一个 block，潜在的浪费 3K。
情况二：文件体积很大（5K)，那么会占用两个（5K-4K 剩下的 1K 也要占用一个 block)。
一个 inode 大小仅为 128bytes（Ext3)，但记录一个 block 则消耗 4bytes，当写 inode 被占满后会取出一个 block 用于号码记录而不再是保存实际的文件系统。2

### 6.4 挂载硬件设备

**挂载操作**指的是当用户需要使用硬盘设备或分区数据时，需要先将其与一个已存在的目录文件**做关联**，而这个动作就叫“**挂载**”。
**mount 命令用于挂载文件系统，格式为：“mount 文件系统 挂载目录”。**
将光盘文件挂载:“mount /dev/cdrom /media/cdrom”。

| 参数 | 作用 |
| --- | --- |
| -a | 挂载所有在/etc/fstab 中定义的文件系统 |
| -t | 指定文件系统的类型 |

如果需要将设备”/dev/sdb2“挂载到”/backup“目录，文件格式为 ext4，该如何操作那?
**执行命令：mount /dev/sdb2 /backup**
很惊讶吗?mount 命令只需要填写设备与挂载目录参数即可，一般来讲系统会自动去判断要挂载文件的类型~
使用 mount 命令执行挂载操作后立即就可以使用该文件系统了，但重启后则失效。如果想让重启后依然生效，我们就必须将挂载信息按照指定的格式写入到**/etc/fstab** 文件中。
“**/etc/fstab**”包含着**文件系统**与**挂载信息**等内容，因为过于重要，所以只有 root 用户才可以编辑它。
**填写格式如下：**“设备文件 挂载目录 格式类型 权限选项 自检 优先级”

---

**设备文件：**一般为设备的路径+名称，也可以写 UUID 值等。
**挂载目录：**指定要挂载到的目录，需挂载前创建好。
**格式类型：**即指定文件系统的格式，比如有 ext3/ext4/xfs/iso9660/swap 等。
**权限选项：**默认为 defaults(rw,suid,dev,exec,auto,nouser,async)，可指定 acl 或 quota 等。
**自检：**若为 1 则开机后进行磁盘自检，0 为不自检。
**优先级：**若“自检”为 1，则可对多块硬盘进行优先级设置。

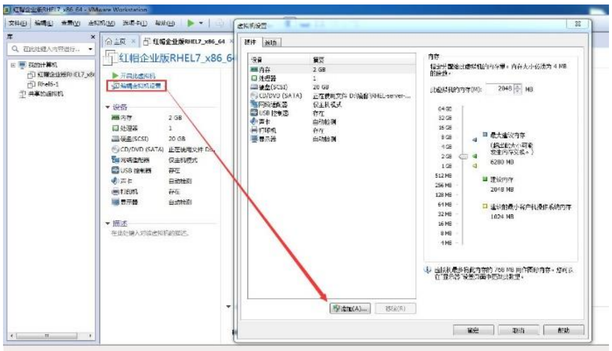
定义设备"/dev/sdb2"开机自动挂载到"/backup"目录，文件格式为 ext4，默认权限且无需开机自检：
正确写法："/dev/sdb2 /backup ext4 defaults 0 0"。
当读者挂载光盘镜像的时候请将文件类型设置为 iso9660,其余设备类型请结合实际情况灵活使用。
**umount 命令用于撤销已经挂载的设备文件，格式为："umount [挂载点/设备文件]"。**
取消对/dev/sdb2 设备文件的挂载：
[root@linuxprobe ~]# umount /dev/sdb2

### 6.5 添加硬盘设备

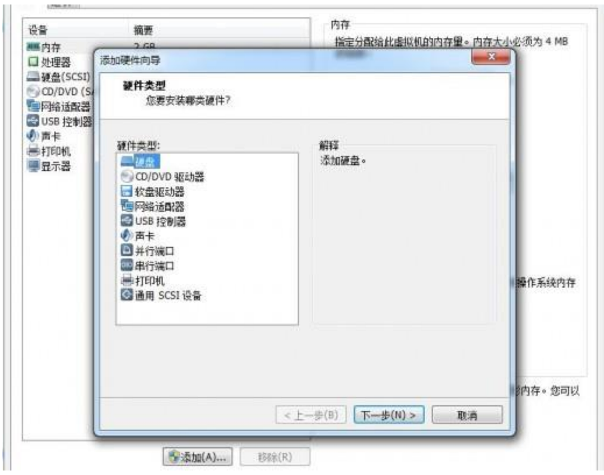
当全新安装了一块新的硬盘设备后，为了更充分、安全的利用硬盘空间首先要进行磁盘的分区，然后格式化，最后挂载使用。

**模拟训练:对新添加的硬盘设备进行分区、格式化并挂载到/newFS 目录。**

第 1 步: 在虚拟机中添加用来做逻辑卷实验的硬盘。




第 2 步：选择磁盘。




第 3 步：选择磁盘类型。



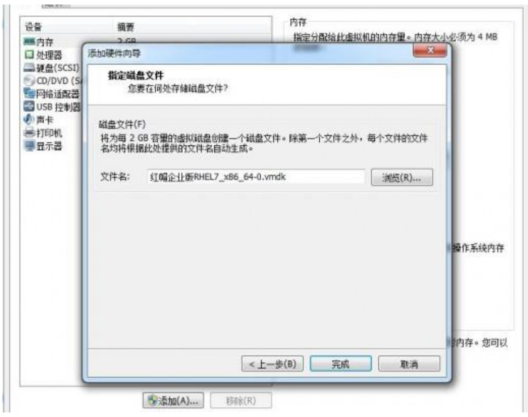

第 4 步：选择创建新的磁盘。



第 5 步：设置磁盘的大小。




第 6 步：默认的磁盘名称即可。




第 7 步：成功添加的硬盘出现在列表中。




**第 2 步:将新添加的硬盘进行分区。**

**fdisk 命令用于管理磁盘分区，格式为：“fdisk [磁盘名称]”。**

管理某硬盘的分区:“fdisk /dev/sda”

| 参数 | 作用 |
|---|---|
| m | 查看全部可用的参数 |
| n | 添加新的分区 |
| d | 删除某个分区信息 |
| l | 列出所有可用的分区类型 |
| t | 改变某个分区的类型 |
| p | 查看分区表信息 |
| w | 保存并退出 |
| q | 不保存直接退出 |

使用 fdisk 命令对 sdb 硬盘进行分区：

[root@linuxprobe ~]# fdisk /dev/sdb

Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.

Be careful before using the write command.

Device does not contain a recognized partition table

Building a new DOS disklabel with disk identifier 0x47d24a34.

敲击字符 p 查看分区表信息（当前为空）：

Command (m for help): p

Disk /dev/sdb: 21.5 GB, 21474836480 bytes, 41943040 sectors

Units = sectors of 1 * 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk label type: dos

Disk identifier: 0x47d24a34

Device Boot Start End Blocks Id System

敲击字符 n 创建新的分区信息：

Command (m for help): n

敲击字符 p，这个 p 代表是主分区，e 为扩展分区：

Partition type:

p primary (0 primary, 0 extended, 4 free)

e extended

Select (default p): p

敲击数字 1 代表分区编号为 1：

Partition number (1-4, default 1): 1

磁盘的起始扇区，直接回车即可：

First sector (2048-41943039, default 2048):

键入+2G，代表该分区的大小为 2G：

Using default value 2048

Last sector, +sectors or +size{K,M,G} (2048-41943039, default 41943039): +2G

Partition 1 of type Linux and of size 2 GiB is set

再看下分区表信息(增加了 sdb1 分区信息)：

Command (m for help): p

Disk /dev/sdb: 21.5 GB, 21474836480 bytes, 41943040 sectors

Units = sectors of 1 * 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk label type: dos

Disk identifier: 0x47d24a34

Device Boot Start End Blocks Id System

/dev/sdb1 2048 4196351 2097152 83 Linux

敲击字符 w，将上述分区信息保存：

Command (m for help): w

The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.

让内核同步分区信息（此步骤仅在没有找到分区设备的情况下才需要执行，非必要动作。）：

[root@linuxprobe ~]# partprobe

**第 3 步:格式化为 xfs 文件系统。**

在 Linux 系统中用于格式化的命令是 mkfs，它支持的文件类型有：

cramfs,ext2,ext3,ext4,fat,msdos,xfs,btrfs,minix,vfat

使用方法非常的简单：”**mkfs.文件类型名称**“，例如要格式分区为 **ext4**，则命令为”**mkfs.ext4 硬盘分区名称**“。

使用 mkfs.xfs 来对/dev/sdb1 进行格式化：

[root@linuxprobe ~]# **mkfs.xfs /dev/sdb1**

**第 4 步:将硬盘设备挂载到/newFS 目录。**

[root@linuxprobe ~]# mkdir /newFS

[root@linuxprobe ~]# mount /dev/sdb1 /newFS/

**第 5 步:设置系统启动后自动挂载该硬盘设备。**

[root@linuxprobe ~]# vim /etc/fstab

**/dev/sdb1 /newFS xfs defaults 0 0**

**第 6 步：查看文件系统的使用情况。**

好棒!我们现在就可以通过访问/newFS 目录来使用硬盘资源啦！另外多教给您几条用于日常了解硬盘使用情况的命令：

**df 命令用于查看挂载点信息与磁盘使用量，格式为：“df [选项] [文件]”。**

查看挂载信息与硬盘使用量: “df -h”

| 参数 | 作用 |
|---|---|
| -a | 显示出所有的文件系统（包括虚拟的） |
| --total | 展出出总体使用量 |
| -h | 更易读的容量格式如 1K,234M,2G… |
| -i | 展示出 Inode 的信息（默认是磁盘使用信息） |
| -T | 显示出文件系统的类型 |

查看到所有已挂载的挂载信息与硬盘使用情况：

[root@linuxprobe ~]# df -h

Filesystem Size Used Avail Use% Mounted on
/dev/mapper/rhel-root 18G 3.5G 15G 20% /
devtmpfs 905M 0 905M 0% /dev
tmpfs 914M 140K 914M 1% /dev/shm
tmpfs 914M 8.8M 905M 1% /run
tmpfs 914M 0 914M 0% /sys/fs/cgroup
/dev/sr0 3.5G 3.5G 0 100% /media/cdrom
/dev/sda1 497M 119M 379M 24% /boot
/dev/sdb1 2.0G 33M 2.0G 2% /newFS

**du 命令用于查看磁盘的使用量，格式为：“ du [选项] [文件]”。**

查看根目录中各文件夹所占空间:du -sh /

查看当前目录下各文件所占空间:du -sh *

| 参数 | 作用 |
| --- | --- |
| -a | 评估每个文件而非目录整体占用量。 |
| -c | 评估每个文件并计算出总占用量总和。 |
| -h | 更易读的容量格式如 1K,234M,2G… |
| -s | 仅显示占用量总和。 |

复制一些文件到新的分区(省略部分复制过程信息)：

[root@linuxprobe ~]# cp /etc/* /newFS/

cp: omitting directory ‘/etc/abrt’

cp: omitting directory ‘/etc/alsa’

cp: omitting directory ‘/etc/festival’

cp: omitting directory ‘/etc/yum’

cp: omitting directory ‘/etc/yum.repos.d’

查看到该挂载目录的占用硬盘量：

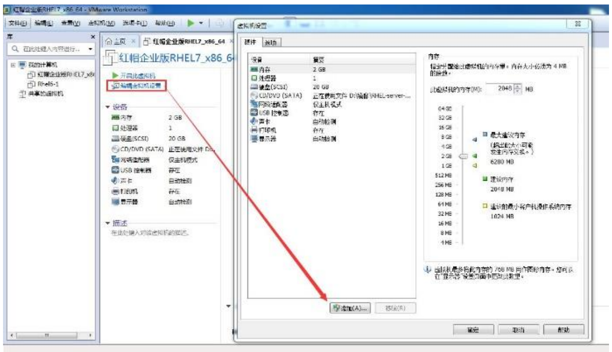
[root@linuxprobe ~]# du -sh /newFS/

1.3M /newFS/

### 6.6 添加交换分区

SWAP 即交换分区是一种类似于 Windows 系统虚拟内存的功能，将一部分硬盘空间虚拟成内存来使用，从而解决内存容量不足的情况，因为 SWAP 毕竟是用硬盘资源虚拟的，所以速度上比真实物理内存要慢很多，一般只有当真实物理内存耗尽时才会调用 SWAP。

第 1 步:在虚拟机中添加用来做逻辑卷实验的硬盘。



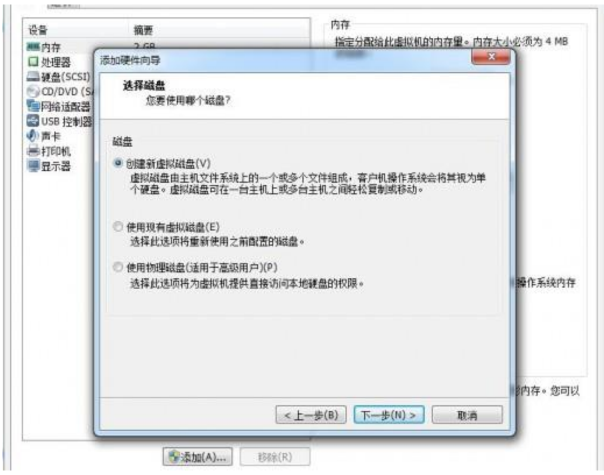
第 2 步：选择磁盘。




第 3 步：选择磁盘类型。



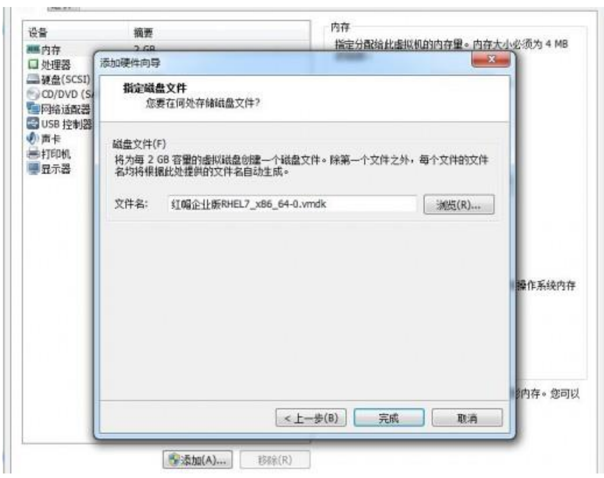

第 4 步：选择创建新的磁盘。




第 5 步：设置磁盘的大小。




第 6 步：默认的磁盘名称即可。




第 7 步：成功添加的硬盘出现在列表中。




**第 2 步:将新添加的硬盘进行分区。**

对新添加的硬盘设备分区：

[root@linuxprobe ~]# fdisk /dev/sdb

Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.

Be careful before using the write command.

Device does not contain a recognized partition table

Building a new DOS disklabel with disk identifier 0xb3d27ce1.

创建新的分区信息：

Command (m for help): n

Partition type:

p primary (0 primary, 0 extended, 4 free)

e extended

类型为主分区：

Select (default p): p

分区号为 1。

Partition number (1-4, default 1): 1

其实扇区直接敲击回车，默认为 2048 即可：

First sector (2048-41943039, default 2048):

Using default value 2048

结束扇区部分输入+5G，设置分区大小为 5G：

Last sector, +sectors or +size{K,M,G} (2048-41943039, default 41943039): +5G

Partition 1 of type Linux and of size 5 GiB is set

修改分区的类型：

Command (m for help): t

Selected partition 1

查看可用的分区类型：

Hex code (type L to list all codes): L

输入 82 代表 swap 分区：

Hex code (type L to list all codes): 82

Changed type of partition 'Linux' to 'Linux swap / Solaris'

再次查看分区表信息（已有分区信息）：

Command (m for help): p
Disk /dev/sdb: 21.5 GB, 21474836480 bytes, 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0xb3d27ce1
Device Boot Start End Blocks Id System
/dev/sdb1 2048 10487807 5242880 82 Linux swap / Solaris
保存分区表的设置：
Command (m for help): w
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.

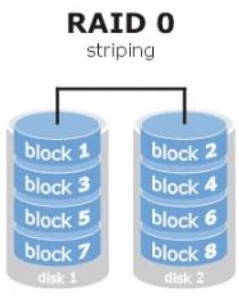
让内核同步分区信息（此步骤仅在没有找到分区设备的情况下才需要执行，非必要动作。）：
[root@linuxprobe ~]# partprobe
将 sdb1 分区设备格式化为 swap 类型：
[root@linuxprobe ~]# mkswap /dev/sdb1
Setting up swapspace version 1, size = 5242876 KiB
no label, UUID=2972f9cb-17f0-4113-84c6-c64b97c40c75
查看当前的内存使用量情况，SWAP 大小为 2047：
[root@linuxprobe ~]# free -m
total used free shared buffers cached
Mem: 1483 782 701 9 0 254
-/+ buffers/cache: 526 957
Swap: 2047 0 2047
将 sdb1 的 SWAP 分区启用：
[root@linuxprobe ~]# swapon /dev/sdb1
再次查看当前系统的内存使用量情况（此时 SWAP 为 7167m）：
[root@linuxprobe ~]# free -m
total used free shared buffers cached
Mem: 1483 785 697 9 0 254
-/+ buffers/cache: 530 953
Swap: 7167 0 7167
设置为开机后自动挂载该 SWAP 分区设备：
[root@linuxprobe ~]# vim /etc/fstab
/dev/sdb1 swap swap defaults 0 0

6.7 磁盘冗余阵列

1988 年由加利福尼亚大学伯克利分校发表的文章首次提到并定义了 RAID，当今 CPU 性能每年可提升 30%-50%但硬盘仅提升 7%，渐渐的已经成为计算机整体性能的瓶颈，并且为了避免硬盘的突然损坏导致数据丢失还加入了冗余备份机制。

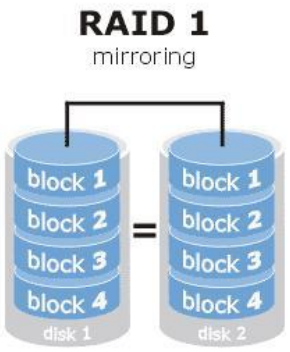
RAID 的早先设计理念为”redundant array of Inexpensive disks“即不贵的硬盘组，而现在的定义是”Redundant Array ofIndependent Disks“即独立的硬盘组，**作用是防止硬盘物理损坏以及增加存储设备的吞吐量**。RAID 常见的组合有 0、1、5 和 10：

RAID0:需要至少两块(含)硬盘，可以有效的提高硬盘的性能和吞吐量，但没有数据的冗余和错误修复能力。



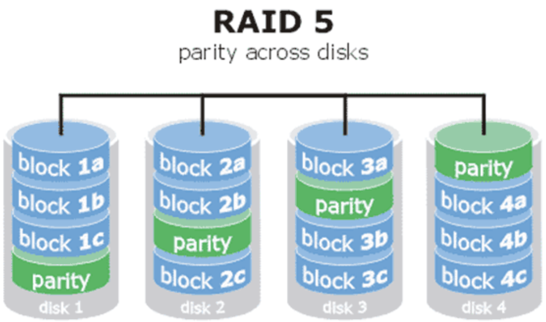

将多块硬盘通过硬件或软件的方式串联在一起，成为一个大的卷集，将数据依次写入到各个硬盘中，这样性能会极大提升，但若任意一块硬盘故障则整个系统的数据都会受到破坏。

RAID1:需要至少两块(含)硬盘，可以有效的提高数据资料的安全性和可修复性，但成本却提高了。




实现原来是在数据写入硬盘时也会在另外一块闲置的硬盘上生成镜像文件，在不影响性能的情况下最大限度保证数据资料的可靠性，只要在一对镜像盘中还有一块硬盘可以使用，那么数据也不会丢失，具有很好的硬盘冗余能力，虽然对数据来讲绝对的安全，但成本却明显增加，磁盘利用率仅为 50%。

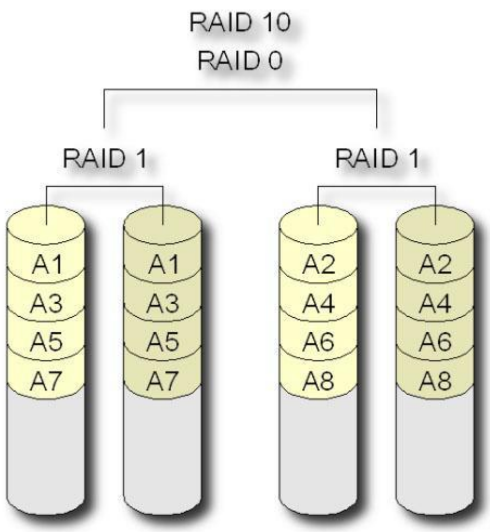
RAID5:需要至少三块(含)硬盘，兼顾存储性能、数据安全和储存成本。




如上图所示”parity”块中保存的是其他硬盘数据的奇偶校验信息（并非其他硬盘的数据），以数据的奇偶校验信息来保证数据的安全，RAID5 不以单独的硬盘来存放数据的奇偶校验信息，而是保存在各个磁盘上。

这样当任何一个硬盘损坏都可以根据其他硬盘上的奇偶校验信息来尝试重建损坏的数据，性能也很高，兼顾了存储性能、数据安全和存储成本，可以看作是 RAID0 与 RAID1 的折中方案。

RAID10:需要至少四块（含）硬盘，兼具速度和安全性，但成本很高。




继承了 RAID0 的快速与 RAID1 的安全，RAID1 在这里提供了冗余备份的阵列，而 RAID0 则负责数据的读写阵列。因这种结构的成本高，一般用于存放要求速度与差错控制的数据。

**mdadm 命令用于管理系统软件 RAID 硬盘阵列，格式为："mdadm [模式] <RAID 设备名称> [选项] [成员设备名称]"。**

**mdadm 管理 RAID 阵列的动作有：**

| 名称 | 作用 |
|---|---|
| Assemble | 将设备加入到以前定义的阵列 |
| Build | 创建一个没有超级块的阵列 |
| Create | 创建一个新的阵列，每个设备具有超级块。 |
| Manage | 管理阵列(如添加和删除)。 |
| Misc | 允许单独对阵列中的某个设备进行操作（如停止阵列）。 |
| Follow or Monitor | 监控状态。 |
| Grow | 改变阵列的容量或设备数目。 |

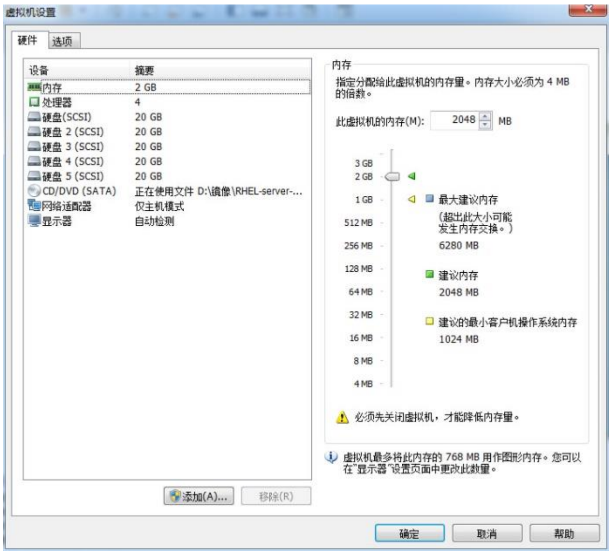
**mdadm 管理 RAID 阵列的参数有：**

| 参数 | 作用 |
|---|---|
| -a | 检测设备名称 |
| -n | 指定设备数量 |
| -l | 指定 raid 级别 |

| -C | 创建 |
| --- | --- |
| -v | 显示过程 |
| -f | 模拟设备损坏 |
| -r | 移除设备 |
| -a | 添加设备 |
| -Q | 查看摘要信息 |
| -D | 查看详细信息 |
| -S | 停止阵列 |

**模拟训练:RAID10 配置流程:**

**第 1 步:在虚拟机中再添加 4 块硬盘:**




**第 2 步:使用 mdadm 命令创建 RAID10,名称为”/dev/md0”。**

-C 代表创建操作，-v 显示创建过程，-a yes 检查 RAID 名称，-n 是用到的硬盘个数，-l 是定义 RAID 的级别而后面写上要加入阵列的硬盘名称。

```
[root@linuxprobe ~]#mdadm -Cv /dev/md0 -a yes -n 4 -l 10 /dev/sdb /dev/sdc /dev/sdd /dev/sde
    mdadm: layout defaults to n2
    mdadm: layout defaults to n2
    mdadm: chunk size defaults to 512K
    mdadm: size set to 20954624K
    mdadm: Defaulting to version 1.2 metadata
    mdadm: array /dev/md0 started.
```

**第 3 步:格式化并挂载使用**

将 RAID 磁盘阵列格式化为 ext4 格式:

```
[root@linuxprobe ~]# mkfs.ext4 /dev/md0
    mke2fs 1.42.9 (28-Dec-2013)
    Filesystem label=
    OS type: Linux
    Block size=4096 (log=2)
    Fragment size=4096 (log=2)
    Stride=128 blocks, Stripe width=256 blocks
    2621440 inodes, 10477312 blocks
    523865 blocks (5.00%) reserved for the super user
    First data block=0
    Maximum filesystem blocks=2157969408
    320 block groups
    32768 blocks per group, 32768 fragments per group
    8192 inodes per group
    Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
    4096000, 7962624
    Allocating group tables: done
    Writing inode tables: done
    Creating journal (32768 blocks): done
    Writing superblocks and filesystem accounting information: done
```

创建挂载目录:

```
[root@linuxprobe ~]# mkdir /RAID
```

进行文件系统的挂载:

```
[root@linuxprobe ~]# mount /dev/md0 /RAID
```

查看磁盘挂载信息:

```
[root@linuxprobe ~]# df -h
    Filesystem Size Used Avail Use% Mounted on
    /dev/mapper/rhel-root 18G 3.0G 15G 17% /
    devtmpfs 905M 0 905M 0% /dev
    tmpfs 914M 84K 914M 1% /dev/shm
    tmpfs 914M 8.9M 905M 1% /run
    tmpfs 914M 0 914M 0% /sys/fs/cgroup
    /dev/sr0 3.5G 3.5G 0 100% /media/cdrom
    /dev/sda1 497M 119M 379M 24% /boot
    /dev/md0 40G 49M 38G 1% /RAID
```

将此磁盘阵列挂载信息设置为重启后也依然生效:

```
[root@linuxprobe ~]# echo "/dev/md0 /RAID ext4 defaults 0 0" >> /etc/fstab
```

**第 4 步:查看/dev/md0 设备信息**

参数-D 查看 RAID 阵列的详细信息:

```
[root@linuxprobe ~]# mdadm -D /dev/md0
    /dev/md0:
    Version : 1.2
```

Creation Time : Tue May 5 07:43:26 2015
Raid Level : raid10
Array Size : 41909248 (39.97 GiB 42.92 GB)
Used Dev Size : 20954624 (19.98 GiB 21.46 GB)
Raid Devices : 4
Total Devices : 4
Persistence : Superblock is persistent
Update Time : Tue May 5 07:46:59 2015
State : clean
Active Devices : 4
Working Devices : 4
Failed Devices : 0
Spare Devices : 0
Layout : near=2
Chunk Size : 512K
Name : localhost.localdomain:0 (local to host localhost.localdomain)
UUID : cc9a87d4:1e89e175:5383e1e8:a78ec62c
Events : 17
Number Major Minor RaidDevice State
0 8 16 0 active sync /dev/sdb
1 8 32 1 active sync /dev/sdc
2 8 48 2 active sync /dev/sdd
3 8 64 3 active sync /dev/sde

**第 5 步:模拟有 1 块硬盘损坏的情况**

使用 mdadm 的-f 参数将/dev/sdb 移出阵列：

[root@linuxprobe ~]# mdadm /dev/md0 -f /dev/sdb
mdadm: set /dev/sdb faulty in /dev/md0

再看下阵列的状态(此时的/dev/sdb 状态被是移除，失败状态)：

[root@linuxprobe ~]# mdadm -D /dev/md0
/dev/md0:
Version : 1.2
Creation Time : Fri May 8 08:11:00 2015
Raid Level : raid10
Array Size : 41909248 (39.97 GiB 42.92 GB)
Used Dev Size : 20954624 (19.98 GiB 21.46 GB)
Raid Devices : 4
Total Devices : 4
Persistence : Superblock is persistent
Update Time : Fri May 8 08:27:18 2015
State : clean, degraded
Active Devices : 3
Working Devices : 3
Failed Devices : 1
Spare Devices : 0

Layout : near=2
Chunk Size : 512K
Name : linuxprobe.com:0 (local to host linuxprobe.com)
UUID : f2993bbd:99c1eb63:bd61d4d4:3f06c3b0
Events : 21
Number Major Minor RaidDevice State
0 0 0 0 removed
1 8 32 1 active sync /dev/sdc
2 8 48 2 active sync /dev/sdd
3 8 64 3 active sync /dev/sde
0 8 16 - faulty /dev/sdb

**第 6 步:损坏后依然正常使用**

因为 RAID10 级别能够允许一组 RAID1 硬盘中存在一个故障盘而不影响使用，所以依然可以正常的创建或删除文件~
现在就把新的硬盘添加进去吧，当然也可以让硬盘 sdb 恢复使用:请**重启后**执行“ **mdadm /dev/md0 -a /dev/sdb**”。

**第 7 步:设置冗余备份磁盘**

现在发现了一个问题没？运维人员需要在硬盘硬件出现故障后手工添加新的磁盘进去，这样会不会比较不方便？
假如初始化 RAID5 阵列时直接给予 4 块硬盘，其中 1 块硬盘设备用于在阵列某块磁盘故障时自动的替换上去，这样很棒吧！
先将磁盘系统卸载：
[root@linuxprobe ~]# umount /dev/md0
停止该阵列设备，彻底的停用：
[root@linuxprobe ~]# mdadm -S /dev/md0
mdadm: stopped /dev/md0
现在该阵列已经找不到了：
[root@linuxprobe ~]# mdadm -D /dev/md0
mdadm: cannot open /dev/md0: No such file or directory
创建 RAID5 并设置 1 块备份故障盘：
[root@linuxprobe ~]# mdadm -Cv /dev/md0 -n 3 -l 5 -x 1 /dev/sdb /dev/sdc /dev/sdd /dev/sde
mdadm: layout defaults to left-symmetric
mdadm: layout defaults to left-symmetric
mdadm: chunk size defaults to 512K
mdadm: /dev/sdb appears to be part of a raid array:
level=raid10 devices=4 ctime=Fri May 8 08:11:00 2015
mdadm: /dev/sdc appears to be part of a raid array:
level=raid10 devices=4 ctime=Fri May 8 08:11:00 2015
mdadm: /dev/sdd appears to be part of a raid array:
level=raid10 devices=4 ctime=Fri May 8 08:11:00 2015
mdadm: /dev/sde appears to be part of a raid array:
level=raid10 devices=4 ctime=Fri May 8 08:11:00 2015
mdadm: size set to 20954624K
此处需要输入 y，确认创建这个阵列：
Continue creating array? y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
查看下阵列的详细信息(Spare Devices 数量为 1)：

```
[root@linuxprobe ~]# mdadm -D /dev/md0
/dev/md0:
Version : 1.2
Creation Time : Fri May 8 09:20:35 2015
Raid Level : raid5
Array Size : 41909248 (39.97 GiB 42.92 GB)
Used Dev Size : 20954624 (19.98 GiB 21.46 GB)
Raid Devices : 3
Total Devices : 4
Persistence : Superblock is persistent
Update Time : Fri May 8 09:22:22 2015
State : clean
Active Devices : 3
Working Devices : 4
Failed Devices : 0
Spare Devices : 1
Layout : left-symmetric
Chunk Size : 512K
Name : linuxprobe.com:0 (local to host linuxprobe.com)
UUID : 44b1a152:3f1809d3:1d234916:4ac70481
Events : 18
Number Major Minor RaidDevice State
0 8 16 0 active sync /dev/sdb
1 8 32 1 active sync /dev/sdc
4 8 48 2 active sync /dev/sdd
3 8 64 - spare /dev/sde
```

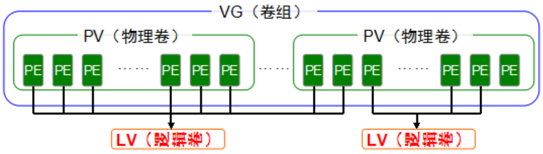
将磁盘阵列格式化为 ext4 系统：

```
[root@linuxprobe ~]# mkfs.ext4 /dev/md0
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=128 blocks, Stripe width=256 blocks
2621440 inodes, 10477312 blocks
523865 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=2157969408
320 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
4096000, 7962624
```

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
因为前面设置过 fstab 文件，所以现在可以直接给挂载：
[root@linuxprobe ~]# mount -a
将/dev/sdb 设备设置为故障并移出阵列：
[root@linuxprobe ~]# mdadm /dev/md0 -f /dev/sdb
mdadm: set /dev/sdb faulty in /dev/md0
再来看下阵列的详细信息(此时硬盘 sde 直接顶替上去了)：
[root@linuxprobe ~]# mdadm -D /dev/md0
/dev/md0:
Version : 1.2
Creation Time : Fri May 8 09:20:35 2015
Raid Level : raid5
Array Size : 41909248 (39.97 GiB 42.92 GB)
Used Dev Size : 20954624 (19.98 GiB 21.46 GB)
Raid Devices : 3
Total Devices : 4
Persistence : Superblock is persistent
Update Time : Fri May 8 09:23:51 2015
State : active, degraded, recovering
Active Devices : 2
Working Devices : 3
Failed Devices : 1
Spare Devices : 1
Layout : left-symmetric
Chunk Size : 512K
Rebuild Status : 0% complete
Name : linuxprobe.com:0 (local to host linuxprobe.com)
UUID : 44b1a152:3f1809d3:1d234916:4ac70481
Events : 21
Number Major Minor RaidDevice State
3 8 64 0 spare rebuilding /dev/sde
1 8 32 1 active sync /dev/sdc
4 8 48 2 active sync /dev/sdd
0 8 16 - faulty /dev/sdb

6.8 逻辑卷管理器

当用户根据实际情况需要对分区增加、减小等调整时，经常会受到硬盘“灵活性”的限制，很不方便。
逻辑卷管理器则是在磁盘分区与文件系统之间添加的逻辑层，提供一个抽象的卷组，使得管理者可以忽略底层磁盘布局，从而实现对分区的灵活动态调整，这毫不夸张，所以红帽 RHEL7 系统已经默认启用了 LVM(Logical Volume Manager)机制。

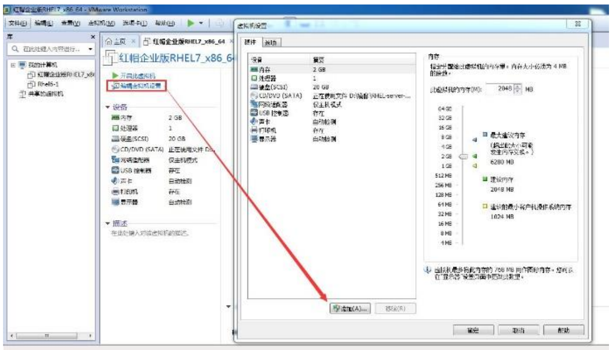



**物理卷**（PV,Physical Volume）: 整个硬盘设备或使用 fdisk 命令建立的硬盘分区。
**卷组**（VG,Volume Group） ：由一个或多个物理卷（PV）组成的整体
**逻辑卷**（LV,Logical Volume） ：从卷组（VG）出切割出的空间来用于创建文件系统，大小由 PE 的个数决定。
基本单元（PE,Physical Extent）默认为 4MB 的基本块。

| 功能/命令 | 物理卷管理 | 卷组管理 | 逻辑卷管理 |
|---|---|---|---|
| 扫描 | pvscan | vgscan | lvscan |
| 建立 | pvcreate | vgcreate | lvcreate |
| 显示 | pvdisplay | vgdisplay | lvdisplay |
| 删除 | pvremove | vgremove | lvremove |
| 扩展 | | vgextend | lvextend |

**模拟训练 A:创建一个容量为 150M 的逻辑卷 vo，格式化为 XFS 并挂载到/mnt/xfs。**

第 1 步:在虚拟机中添加 1 块用来做逻辑卷实验的硬盘



第 2 步：选择磁盘。



第 3 步：选择磁盘类型。




第 4 步：选择创建新的磁盘。




第 5 步：设置磁盘的大小。




第 6 步：默认的磁盘名称即可。




第 7 步：成功添加的硬盘出现在列表中。




第 2 步:创建一个大小为 300M 的分区(sdb1),标签为 lvm:

对硬盘 sdb 进行分区:

[root@linuxprobe ~]# fdisk /dev/sdb

Device does not contain a recognized partition table

创建新的分区：

Command (m for help): n

类型为主分区：

Partition type:

p primary (0 primary, 0 extended, 4 free)

e extended

Select (default p): p

分区号为 1：

Partition number (1-4, default 1): 1

直接写“+300M”即可生成大小为 300M 的分区：

First sector (2048-41943039, default 2048):

Using default value 2048

Last sector, +sectors or +size{K,M,G} (2048-41943039, default 41943039): +300M

Partition 1 of type Linux and of size 300 MiB is set

查看下分区信息：

Command (m for help): p

Device Boot Start End Blocks Id System

/dev/sdb1 2048 616447 307200 83 Linux

修改分区类型：

Command (m for help): t

Selected partition 1

修改分区类型为 lvm（代码是 8e)：

Hex code (type L to list all codes): 8e

Changed type of partition 'Linux' to 'Linux LVM'

再看下分区信息：

Command (m for help): p

Device Boot Start End Blocks Id System

/dev/sdb1 2048 309247 153600 8e Linux LVM

确认无误，写入分区表信息：

Command (m for help): w

The partition table has been altered!

Calling ioctl() to re-read partition table.

Syncing disks.

让内核同步分区信息（此步骤仅在没有找到分区设备的情况下才需要执行，非必要动作。)：

[root@linuxprobe ~]# partprobe

**第 3 步:启用 LVM 并创建 vo 逻辑卷并格式化为 ext4 格式**

将新建的分区设置为物理卷：

[root@linuxprobe ~]# pvcreate /dev/sdb1

Physical volume "/dev/sdb1" successfully created

将物理卷加入卷组：

[root@linuxprobe ~]# vgcreate rhcsa /dev/sdb1

Volume group "rhcsa" successfully created

查看卷组信息:

[root@linuxprobe ~]# vgdisplay

--- Volume group ---

VG Size 296.00 MiB

PE Size 4.00 MiB

Total PE 74

Alloc PE / Size 0 / 0

Free PE / Size 74 / 296.00 MiB

VG UUID 8hLPQU-Tc6f-PMsa-4tq5-iT0p-vSbI-sOafqG

生成大小为 37 个 PE 的逻辑卷（37*4MiB 为 148M）:

[root@linuxprobe ~]# lvcreate -n vo -l 37 rhcsa

Logical volume "vo" created

格式化为 ext4:

[root@linuxprobe ~]# mkfs.ext4 /dev/rhcsa/vo

Writing superblocks and filesystem accounting information: done

创建一个名为/rhcsa 的目录用于挂载该逻辑卷。

[root@linuxprobe ~]# mkdir /rhcsa

挂载硬盘设备:

[root@linuxprobe ~]# mount /dev/rhcsa/vo /rhcsa

查看挂载信息（rhcsa-vo 为 140M 是合理取值）:

[root@linuxprobe ~]# df -h

Filesystem Size Used Avail Use% Mounted on

/dev/mapper/rhel-root 18G 3.0G 15G 17% /

devtmpfs 905M 0 905M 0% /dev

tmpfs 914M 140K 914M 1% /dev/shm

tmpfs 914M 8.9M 905M 1% /run

tmpfs 914M 0 914M 0% /sys/fs/cgroup

/dev/sda1 497M 119M 379M 24% /boot

/dev/sr0 3.5G 3.5G 0 100% /run/media/root/RHEL-7.0 Server.x86_64

/dev/mapper/rhcsa-vo 140M 1.6M 128M 2% /rhcsa

**模拟训练 B：将上个实验中的逻辑卷 vo 容量扩展到 290M。**

若要对 LVM 进行调整，一定要先卸载:

[root@linuxprobe ~]# umount /rhcsa

将逻辑卷扩展到 290M:

[root@linuxprobe ~]# lvextend -L 290M /dev/rhcsa/vo

Rounding size to boundary between physical extents: 292.00 MiB

Extending logical volume vo to 292.00 MiB

Logical volume vo successfully resized

检查磁盘完整性，重置硬盘容量:

[root@linuxprobe ~]# e2fsck -f /dev/rhcsa/vo

/dev/rhcsa/vo: 11/38000 files (0.0% non-contiguous), 10453/151552 blocks\

[root@linuxprobe ~]# resize2fs /dev/rhcsa/vo

Resizing the filesystem on /dev/rhcsa/vo to 299008 (1k) blocks.

The filesystem on /dev/rhcsa/vo is now 299008 blocks long.
重新挂载硬盘设备：
[root@linuxprobe ~]# mount /dev/rhcsa/vo /rhcsa
看到挂载信息（当前逻辑卷大小已为 279M）：
[root@linuxprobe ~]# df -h
Filesystem Size Used Avail Use% Mounted on
/dev/mapper/rhel-root 18G 3.0G 15G 17% /
devtmpfs 905M 0 905M 0% /dev
tmpfs 914M 140K 914M 1% /dev/shm
tmpfs 914M 8.9M 905M 1% /run
tmpfs 914M 0 914M 0% /sys/fs/cgroup
/dev/sda1 497M 119M 379M 24% /boot
/dev/sr0 3.5G 3.5G 0 100% /run/media/root/RHEL-7.0 Server.x86_64
/dev/mapper/rhcsa-vo 279M 2.1M 259M 1% /rhcsa

**模拟训练 C：将上个实验中的逻辑卷 vo 容量减小到 120M。**

卸载文件系统：
[root@linuxprobe ~]# umount /rhcsa
检查文件系统的完整性：
[root@linuxprobe ~]# e2fsck -f /dev/rhcsa/vo
e2fsck 1.42.9 (28-Dec-2013)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/rhcsa/vo: 11/74000 files (0.0% non-contiguous), 15507/299008 blocks
将逻辑卷的减小到 120M:
[root@linuxprobe ~]# resize2fs /dev/rhcsa/vo 120M
resize2fs 1.42.9 (28-Dec-2013)
Resizing the filesystem on /dev/rhcsa/vo to 122880 (1k) blocks.
The filesystem on /dev/rhcsa/vo is now 122880 blocks long.
使用 lvreduce 命令将文件系统调整为 120M:
[root@linuxprobe ~]# lvreduce -L 120M /dev/rhcsa/vo
WARNING: Reducing active logical volume to 120.00 MiB
THIS MAY DESTROY YOUR DATA (filesystem etc.)
Do you really want to reduce vo? [y/n]: y
Reducing logical volume vo to 120.00 MiB
Logical volume vo successfully resized
重新挂载文件系统:
[root@linuxprobe ~]# mount /dev/rhcsa/vo /rhcsa
查看挂载信息，(逻辑卷已经变成 113M)：
[root@linuxprobe ~]# df -h
Filesystem Size Used Avail Use% Mounted on
/dev/mapper/rhel-root 18G 3.7G 14G 21% /

devtmpfs 734M 0 734M 0% /dev

tmpfs 742M 140K 742M 1% /dev/shm

tmpfs 742M 8.8M 734M 2% /run

tmpfs 742M 0 742M 0% /sys/fs/cgroup

/dev/sr0 3.5G 3.5G 0 100% /media/cdrom

/dev/sda1 497M 119M 379M 24% /boot

/dev/mapper/rhcsa-vo 113M 1.6M 103M 2% /rhcsa

**模拟训练 D：使用逻辑卷快照功能**

LVM 的逻辑卷快照功能可以将逻辑卷的数据保存为备份、以及快速的数据恢复。

查看到逻辑卷详细信息（容量共计 296M，已用 120M，剩余 196M）：

[root@linuxprobe ~]# vgdisplay

--- Volume group ---

VG Name rhcsa

System ID

Format lvm2

Metadata Areas 1

Metadata Sequence No 4

VG Access read/write

VG Status resizable

MAX LV 0

Open LV 1

Max PV 0

Cur PV 1

Act PV 1

VG Size 296.00 MiB

PE Size 4.00 MiB

Total PE 74

Alloc PE / Size 30 / 120.00 MiB

Free PE / Size 44 / 176.00 MiB

VG UUID QxBS5f-beVv-FJnu-GKyu-UWWF-JS8x-ytiAN9

创建原始文件，写入一行文字：

[root@linuxprobe ~]# echo "Welcome to Linuxprobe.com" > /rhcsa/readme.txt

[root@linuxprobe ~]# ls /rhcsa

lost+found readme.txt

对 rhcsa 卷组的 vo 逻辑卷做一个名称为 SNAP 而大小为 150M 的逻辑卷快照：

[root@linuxprobe ~]# lvcreate -L 150M -s -n SNAP /dev/rhcsa/vo

Rounding up size to full physical extent 152.00 MiB

Reducing COW size 152.00 MiB down to maximum usable size 124.00 MiB.

Logical volume "SNAP" created

查看逻辑卷和快照的信息：

[root@linuxprobe ~]# lvs

LV VG Attr LSize Pool Origin Data% Move Log Cpy%Sync Convert

SNAP rhcsa swi-a-s--- 124.00m vo 0.01

vo rhcsa owi-aos--- 120.00m

root rhel -wi-ao---- 17.51g
swap rhel -wi-ao---- 2.00g
在逻辑卷中创建一个 100M 的文件:
[root@linuxprobe ~]# dd if=/dev/zero of=/rhcsa/files count=1 bs=100M
1+0 records in
1+0 records out
104857600 bytes (105 MB) copied, 1.31474 s, 79.8 MB/s
再来看下逻辑卷快照的使用量:
[root@linuxprobe ~]# lvs
LV VG Attr LSize Pool Origin Data% Move Log Cpy%Sync Convert
SNAP rhcsa swi-a-s--- 124.00m vo 89.76
vo rhcsa owi-aos--- 120.00m
root rhel -wi-ao---- 17.51g
swap rhel -wi-ao---- 2.00g
将文件系统卸载:
[root@linuxprobe ~]# umount /rhcsa
恢复 SNAP 逻辑卷快照内容:
[root@linuxprobe ~]# lvconvert --merge /dev/rhcsa/SNAP
Merging of volume SNAP started.
vo: Merged: 18.2%
vo: Merged: 100.0%
Merge of snapshot into logical volume vo has finished.
Logical volume "SNAP" successfully removed
快照恢复一次后会被自动删除:
[root@linuxprobe ~]# lvs
LV VG Attr LSize Pool Origin Data% Move Log Cpy%Sync Convert
vo rhcsa -wi-a----- 120.00m
root rhel -wi-ao---- 17.51g
swap rhel -wi-ao---- 2.00

重新挂载文件系统:
[root@linuxprobe ~]# mount /dev/rhcsa/vo /rhcsa
原始的文件还在，但刚刚创建的 100M 大文件被清除了:
[root@linuxprobe ~]# ls /rhcsa
lost+found readme.txt

6.9 磁盘容量配额
如前面介绍章节讲到的类 Unix 系统最初设计理念就让许多人一起使用，多任务的操作系统，但是硬件的资源是固定有限的，如果出现个小破坏份子不断的创建文件或下载电影，那么硬盘空间总有一天会被占满的吧，这时就需要 quota 服务帮助我们为每个用户限制可以使用的硬盘空间，一旦超出预算就不再允许他们使用。
quota 的磁盘配额可以限制用户的硬盘可用空间或最大创建文件数量，并且还有软/硬限制的区别:
软限制:当达到软限制时会提示用户，但允许用户在规定期限内继续使用。
硬限制:当达到硬限制时会提示用户，且强制终止用户的操作。
查看内核是否支持 quota 功能:

[root@linuxprobe ~]# dmesg | grep quota

[ 3.140241] VFS: Disk quotas dquot_6.5.2

查看 quota 程序包是否已经安装：

[root@linuxprobe ~]# rpm -q quota

quota-4.01-11.el7.x86_64

查看 boot 目录是否支持 quota 功能（noquota 表示暂时不支持）：

[root@linuxprobe ~]# mount | grep boot

/dev/sda1 on /boot type xfs (rw,relatime,seclabel,attr2,inode64,noquota)

让/boot 目录支持 quota 功能：

[root@linuxprobe ~]# vim /etc/fstab

UUID=6e97ef8f-51f1-4781-8f1c-0acb9f631b32 /boot xfs defaults,uquota 0 0

重启主机后即可生效：

[root@linuxprobe ~]# reboot

查看 boot 目录是否支持 quota 功能(usrquota 表示已经支持)：

[root@linuxprobe Desktop]# mount | grep boot

/dev/sda1 on /boot type xfs (rw,relatime,seclabel,attr2,inode64,usrquota)

创建两个用于 quota 实验的用户 tom：

[root@linuxprobe ~]# useradd tom

需要允许其他用户对/boot 目录写入文件操作：

[root@linuxprobe ~]# chmod -Rf o+w /boot

**xfs_quota 命令用于管理 XFS 文件系统的 quota 硬盘配额，格式为：“quota [参数] 配额 文件系统”。**

| 参数 | 作用 |
|---|---|
| -c 命令 | 以交换式或参数的形式设置要执行的命令。 |
| -p | 设置提示或报错信息的程序名称，默认为 xfs_quota。 |
| -x | 专家模式，能够对 quota 做更多复杂的配置。 |

使用 xfs_quota 命令设置对 tom 用户在/boot 目录的磁盘配额，具体要求如下：

使用 quota 专家模式限制磁盘软限制为 3m、磁盘硬限制为 6m、文件软限制为 3 个且文件硬限制为 6 个。

[root@linuxprobe ~]# xfs_quota -x -c 'limit bsoft=3m bhard=6m isoft=3 ihard=6 tom' /boot

获取当前/boot 目录上的 quota 配额限制：

[root@linuxprobe ~]# xfs_quota -x -c report /boot

User quota on /boot (/dev/sda1) Blocks

User ID Used Soft Hard Warn/Grace

---------- --------------------------------------------------

root 95084 0 0 00 [--------]

tom 0 3072 6144 00 [--------]

切换至 tom 用户：

[root@linuxprobe ~]# su - tom

正常创建了一个为 5M 的文件：

[tom@linuxprobe ~]$ dd if=/dev/zero of=/boot/tom bs=5M count=1

1+0 records in

1+0 records out

5242880 bytes (5.2 MB) copied, 0.123966 s, 42.3 MB/s
创建 8M 文件时强制终止并报错了：
[tom@linuxprobe ~]$ dd if=/dev/zero of=/boot/tom bs=8M count=1
dd: error writing ‘/boot/tom’: Disk quota exceeded
1+0 records in
0+0 records out
6291456 bytes (6.3 MB) copied, 0.0201593 s, 312 MB/s
查看当前用户的 quota 限制（显示硬盘配额已占满）：
[tom@linuxprobe ~]$ quota
Disk quotas for user tom (uid 1001):
Filesystem blocks quota limit grace files quota limit grace
/dev/sda1 6144* 3072 6144 6days 1 3 6

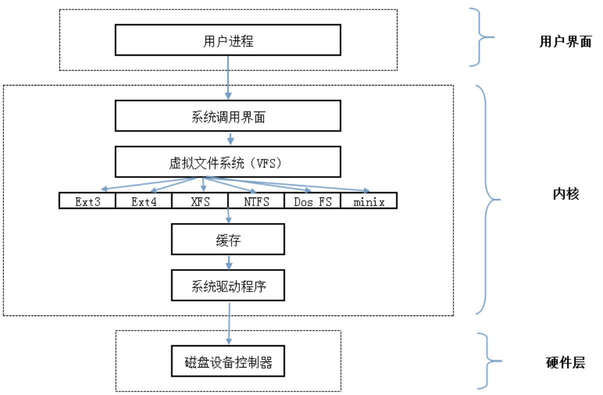
**edquota 命令用于超级用户编辑其他用户的 quota 配额限制，格式为："edquota [参数] [用户]"。**

| 参数 | 作用 |
|---|---|
| -u | 编辑用户的配额限制。 |
| -g | 编辑用户组的配额限制。 |
| -r | 通过 RPC 协议编辑远程的配额。 |

编辑 tom 的配额限制，将硬盘的硬限制修改为 8m(8192k)：
[root@linuxprobe ~]# edquota -u tom
Disk quotas for user tom (uid 1001):
Filesystem blocks soft hard inodes soft hard
/dev/sda1 6144 3072 8192 1 3 6
切换至 tom 用户：
[root@linuxprobe ~]# su - tom
Last login: Mon Sep 7 16:43:12 CST 2015 on pts/0
再来创建 8m 的文件就不会有问题了：
[tom@linuxprobe ~]$ dd if=/dev/zero of=/boot/tom bs=8M count=1
1+0 records in
1+0 records out
8388608 bytes (8.4 MB) copied, 0.0268044 s, 313 MB/s

6.10 虚拟文件系统

随着计算机系统的发展产生出了众多的文件系统，为了使用户在读取或写入文件时不用关心底层的硬盘结构，于是在 Linux 内核中的软件层为用户程序提供了一个文件系统接口(VFS,Virtual File System)，这样就转而统一对这个虚拟文件系统进行操作啦。即**实际文件系统在 VFS 下隐藏了自己的特性和细节，使得我们在日常使用时觉得"文件系统都是一样的"。**



6.11 软硬方式链接

在 Linux 系统中的 ln 命令能够让用户创建出两种不同类型的文件快捷方式，一定要注意区分：

**硬链接(hard link)**可以被理解为一个“指向原始文件 inode 的指针”，系统不为它分配独立的 inode 与文件，**所以实际上来讲硬链接文件与原始文件其实是同一个文件，只是名字不同。**于是每添加一个硬链接，该文件的 inode 连接数就会增加 1，直到该文件的 inode 连接数归 0 才是彻底删除。概括来说因为硬链接实际就是指向原文件 inode 的指针，即便原始文件被删除依然可以通过链接文件访问，但是不能跨文件系统也不能链接目录文件。

**软链接也称为符号链接（symbolic link）**即“仅仅包含它索要链接文件的路径名”因此能做目录链接也可以跨越文件系统，但原始文件被删除后链接文件也将失效，如同 Winodw™中的“快捷方式”。

**ln 命令用于创建链接文件，格式为：“ln [选项] 目标”。**

创建硬链接:“ln 文件名 链接名”

创建软链接:“ln -s 文件名 连接名”

| 参数 | 作用 |
|---|---|
| -s | 创建"符号链接"(默认是硬链接) |
| -f | 强制创建文件或目录的链接 |
| -i | 覆盖前先询问 |
| -v | 显示创建链接的过程 |

对/etc 目录做出一个名为 etc 的软连接。

[root@linuxprobe ~]# ln -s /etc etc

**本章结束，您可以在此写下笔记：**

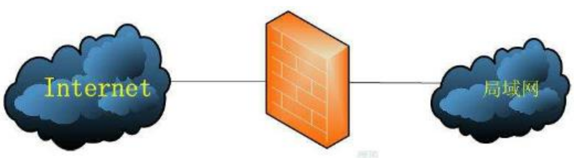
第 7 章 Iptables 与 Firewalld 防火墙。

**章节简述：**

红帽 RHEL7 系统已经用 firewalld 服务替代了 iptables 服务，新的防火墙管理命令 **firewall-cmd** 与图形化工具 **firewall-config**。本章节基于数十个防火墙需求，使用规则策略完整演示对数据包的过滤、SNAT/SDAT 技术、端口转发以及负载均衡等实验。不光光学习 iptables 命令与 firewalld 服务，还新增了 **Tcp_wrappers** 防火墙服务小节，简单配置即可保证系统与服务的安全。

7.1 了解防火墙管理工具

防火墙虽有软件或硬件之分但主要功能还是**依据策略对外部请求进行过滤**，成为公网与内网之间的保护屏障，防火墙会监控每一个数据包并判断是否有相应的匹配策略规则，直到满足其中一条策略规则为止，而防火墙规则策略可以是基于来源地址、请求动作或协议来定制的，**最终仅让合法的用户请求流入到内网中，其余的均被丢弃。**



在红帽 RHEL7 系统中 **firewalld 服务**取代了 **iptables 服务**，但依然可以使用 **iptables 命令**来管理内核的 netfilter。这对于接触 Linux 系统比较早或学习过红帽 RHEL6 系统的读者来讲，突然接触 firewalld 服务会比较抵触，可能会觉得新增 Firewalld 服务是一次不小的改变，其实这样讲也是有道理的。**但其实 Iptables 服务与 Firewalld 服务都不是真正的防火墙，它们都只是用来定义防火墙规则功能的“防火墙管理工具”**，将定义好的规则交由内核中的 **netfilter** 即网络过滤器来读取，从而真正实现防火墙功能，所以其实在配置规则的思路上是**完全一致**的，而我会在本章中将 iptables 命令与 firewalld 服务的使用方法都教授给你们，坦白讲日常工作无论用那种都是可行的。

7.2 Iptables 命令

iptables 命令用于创建数据过滤与 NAT 规则，主流的 Linux 系统都会默认启用 iptables 命令，但其参数较多且规则策略相对比较复杂。

7.2.1 规则链与策略

在 iptables 命令中设置数据过滤或处理数据包的策略叫做**规则**，将多个规则合成一个**链**。

**举例来说:**小区门卫有两条的规则，将这两个规则可以合成一个规则链：

遇到外来车辆需要登记。

严禁快递小哥进入社区。

但是光有策略还不能保证社区的安全，我们需要告诉门卫（iptables）这个策略（规则链）是作用于哪里的，并赋予安保人员可能的操作有这些，如：“允许”，“登记”，“拒绝”，“不理他”，对应到 iptables 命令中则常见的控制类型有：

ACCEPT:允许通过.

LOG:记录日志信息,然后传给下一条规则继续匹配.

REJECT:拒绝通过,必要时会给出提示.

DROP:直接丢弃,不给出任何回应.

其中 REJECT 和 DROP 的操作都是将数据包拒绝，但 REJECT 会再回复一条“您的信息我已收到，但被扔掉了”。

**通过 ping 命令测试 REJECT 情况会是这样的：**

```
[root@localhost ~]# ping -c 2 192.168.10.10
PING 192.168.10.10 (192.168.10.10) 56(84) bytes of data.
From 192.168.10.10 icmp_seq=1 Destination Port Unreachable
From 192.168.10.10 icmp_seq=2 Destination Port Unreachable
--- 192.168.10.10 ping statistics ---
2 packets transmitted, 0 received, +2 errors, 100% packet loss, time 3002ms
```

**但如果是 DROP 则不予响应：**

```
[root@localhost ~]# ping -c 4 192.168.10.10
PING 192.168.10.10 (192.168.10.10) 56(84) bytes of data.

--- 192.168.10.10 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3000ms
```

而规则链则依据处理数据包的位置不同而进行分类：

PREROUTING:在进行路由选择前处理数据包

INPUT:处理入站的数据包

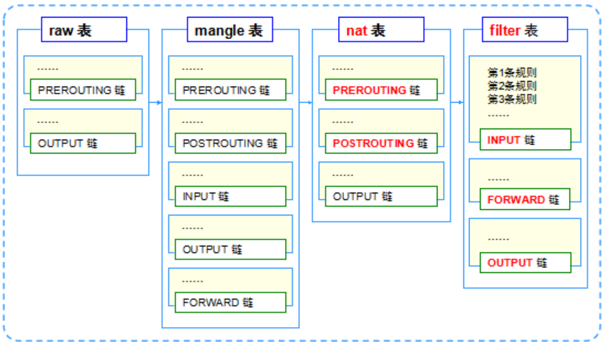
OUTPUT:处理出站的数据包

FORWARD:处理转发的数据包

POSTROUTING:在进行路由选择后处理数据包

Iptables 中的**规则表**是用于容纳**规则链**，规则表默认是允许状态的，那么规则链就是设置被禁止的规则，而反之如果规则表是禁止状态的，那么规则链就是设置被允许的规则。

raw 表:确定是否对该数据包进行状态跟踪
mangle 表:为数据包设置标记
nat 表:修改数据包中的源、目标 IP 地址或端口
filter 表:确定是否放行该数据包（过滤）




**规则表的先后顺序:raw→mangle→nat→filter**

**规则链的先后顺序:**

**入站顺序:**PREROUTING→INPUT
**出站顺序:**OUTPUT→POSTROUTING
**转发顺序:**PREROUTING→FORWARD→POSTROUTING

**还有三点注意事项:**

1.没有指定规则表则默认指 filter 表。
2.不指定规则链则指表内所有的规则链。
3.在规则链中匹配规则时会依次检查，匹配即停止（**LOG 规则例外**），若没匹配项则按链的默认状态处理。

7.2.2 基本的命令参数

**iptables 命令用于管理防火墙的规则策略，格式为：“iptables [-t 表名] 选项 [链名] [条件] [-j 控制类型]”。**
表格为读者总结了几乎所有常用的 iptables 参数，如果记不住也没关系，用时来查就行，看完后来学习下如何组合并使用吧：

| 参数 | 作用 |
| --- | --- |
| -P | 设置默认策略:iptables -P INPUT (DROP\|ACCEPT) |
| -F | 清空规则链 |
| -L | 查看规则链 |
| -A | 在规则链的末尾加入新规则 |
| -I num | 在规则链的头部加入新规则 |

| | |
|---|---|
| -D num | 删除某一条规则 |
| -s | 匹配来源地址 IP/MASK，加叹号"!"表示除这个 IP 外。 |
| -d | 匹配目标地址 |
| -i 网卡名称 | 匹配从这块网卡流入的数据 |
| -o 网卡名称 | 匹配从这块网卡流出的数据 |
| -p | 匹配协议,如 tcp,udp,icmp |
| --dport num | 匹配目标端口号 |
| --sport num | 匹配来源端口号 |

**查看已有的规则：**

[root@linuxprobe ~]# iptables -L

**清空已有的规则：**

[root@linuxprobe ~]# iptables -F

**将 INPUT 链的默认策略设置为拒绝：**

当 INPUT 链默认规则设置为拒绝时，我们需要写入允许的规则策略。

这个动作的目地是当接收到数据包时，按顺序匹配所有的允许规则策略，当全部规则都不匹配时，拒绝这个数据包。

[root@linuxprobe ~]# iptables -P INPUT DROP

**允许所有的 ping 操作：**

[root@linuxprobe ~]# iptables -I INPUT -p icmp -j ACCEPT

**在 INPUT 链的末尾加入一条规则，允许所有未被其他规则匹配上的数据包：**

因为默认规则表就是 filter，所以其中的”**-t filter**“一般省略不写，效果是一样的。

[root@linuxprobe ~]# iptables -t filter -A INPUT -j ACCEPT

**删除上面的那条规则：**

[root@linuxprobe ~]# iptables -D INPUT 2

既然读者已经掌握了 iptables 命令的基本参数，那么来尝试解决模拟训练吧：

**模拟训练 A:**仅允许来自于 192.168.10.0/24 域的用户连接本机的 ssh 服务。

Iptables 防火墙会按照顺序匹配规则，请一定要保证“**允许**”规则是在“**拒绝**”规则的上面。

[root@linuxprobe ~]# iptables -I INPUT -s 192.168.10.0/24 -p tcp --dport 22 -j ACCEPT

[root@linuxprobe ~]# iptables -A INPUT -p tcp --dport 22 -j REJECT

**模拟训练 B:**不允许任何用户访问本机的 12345 端口。

[root@linuxprobe ~]# iptables -I INPUT -p tcp --dport 12345 -j REJECT

[root@linuxprobe ~]# iptables -I INPUT -p udp --dport 12345 -j REJECT

**模拟实验 C(答案模式):**拒绝其他用户从”eno16777736“网卡访问本机 http 服务的数据包。

**答案：**[rootlinuxprobe ~]# iptables -I INPUT -i eno16777736 -p tcp --dport 80 -j REJECT

**模拟训练 D:**禁止用户访问 www.my133t.org。

[root@linuxprobe ~]# iptables -I FORWARD -d www.my133t.org -j DROP

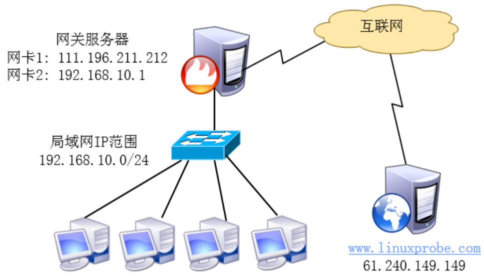
**模拟训练 E:**禁止 IP 地址是 192.168.10.10 的用户上网

[root@linuxprobe ~]# iptables -I FORWARD -s 192.168.10.10 -j DROP

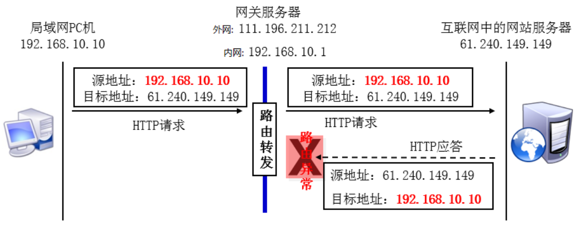
iptables 命令执行后的规则策略仅当前生效，若想重启后依然保存规则需执行”**service iptables save**“。

7.2.3 SNAT 与 DNAT

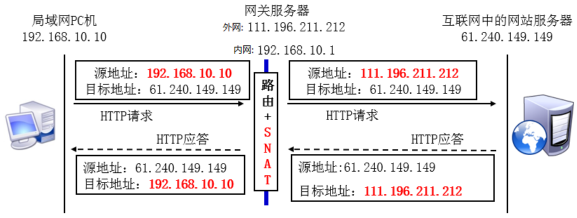
SNAT 即源地址转换技术，能够让多个内网用户通过一个外网地址上网，解决了 IP 资源匮乏的问题，确实很实用。例如读者们来访问《Linux 就该这么学》的网页，则就是通过家中的网关设备（您的无线路由器）进行的 SNAT 转换。




（多用户局域网共享上网的拓扑）

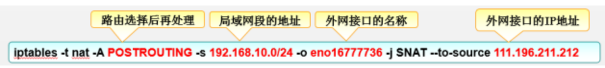



（因未使用 SNAT 技术，所以在网站服务器应答后找不到 192.168.10.10 这台主机，无法正常浏览网页）




**(因使用了 SNAT 地址转换技术，服务器应答后先由网关服务器接收，再分发给内网的用户主机。)**

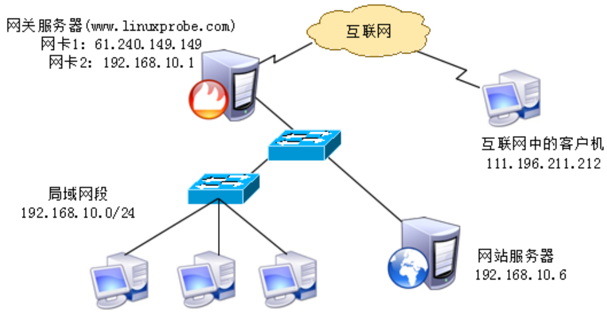
现在需要将"192.168.10.0"网段的内网 IP 用户经过地址转换技术变成外网 IP 地址"111.196.211.212"，这样一来内网 IP 用户就都可以通过这个外网 IP 上网了，使用 iptables 防火墙即可实现 SNAT 源地址转换，根据需求命令如下:



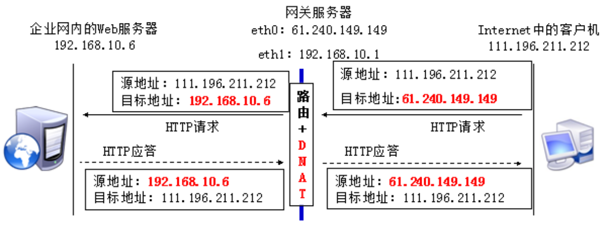

不知读者有无这种经历，当使用联通或者电信上网的时候，每次拨号都会重新分配新的 IP 地址，那么若网关 IP 经常变动怎么办?这种外网 IP 地址不稳定的情况即可使用 MASQUERADE(**动态伪装**):能够自动的寻找外网地址并改为当前正确的外网 IP 地址

iptables -t nat -A POSTROUTING -s 192.168.10.0/24 -j MASQUERADE

DNAT 即目地地址转换技术，则能够让外网 IP 用户访问局域网内不同的服务器。

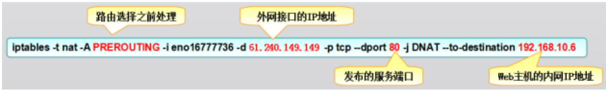



（互联网中的客户机访问网站时的拓扑）




**（DNAT 的数据包转换过程）**

现在希望互联网中的客户机访问到内网”192.168.10.6“这台提供网站服务的主机，那么只需在网关系统上运行这条命令：




7.2.4 端口转发与流量均衡

**端口转发功能可以将原本到某端口的数据包转发到其他端口：**

firewall-cmd --permanent --zone=<区域> --add-forward-port=port=<源端口号>:proto=<协议>:toport=<目标端口号>:toaddr=<目标 IP 地址>

**将访问 192.168.10.10 主机 888 端口的请求转发至 22 端口：**

[root@linuxprobe ~]# firewall-cmd --permanent --zone=**public** --add-forward-port=port=**888**:proto=**tcp**:toport=**22**:toaddr=**192.168.10.10**

success

使用客户机的 ssh 命令访问 192.168.10.10 主机的 888 端口：

[root@linuxprobe ~]# ssh -p 888 192.168.10.10

The authenticity of host '[192.168.10.10]:888 ([192.168.10.10]:888)' can't be established.

ECDSA key fingerprint is b8:25:88:89:5c:05:b6:dd:ef:76:63:ff:1a:54:02:1a.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '[192.168.10.10]:888' (ECDSA) to the list of known hosts.

root@192.168.10.10's password:

Last login: Sun Jul 19 21:43:48 2015 from 192.168.10.10

另外流量均衡技术也是常用的技术，比如将一台主机作为网站的前端服务器，将访问流量分流至内网中 3 台不同的主机上。

iptables -A PREROUTING -i eth0 -p tcp --dport 80 -m state --state NEW -m nth --counter 0 --every 3 --packet 0 -j DNAT --to-destination 192.168.10.10:80

iptables -A PREROUTING -i eth0 -p tcp --dport 80 -m state --state NEW -m nth --counter 0 --every 3 --packet 0 -j DNAT --to-destination 192.168.10.11:80

iptables -A PREROUTING -i eth0 -p tcp --dport 80 -m state --state NEW -m nth --counter 0 --every 3 --packet 0 -j DNAT --to-destination 192.168.10.12:80

7.3 Firewalld 防火墙

**Firewalld 服务**是红帽 RHEL7 系统中默认的防火墙管理工具，特点是拥有**运行时配置**与**永久配置**选项且能够支持动态更新以及"zone"的区域功能概念，使用图形化工具 **firewall-config** 或文本管理工具 **firewall-cmd**，下面实验中会讲到~

7.3.1 区域概念与作用

防火墙的网络区域定义了网络连接的可信等级，我们可以根据不同场景来调用不同的 firewalld 区域，区域规则有：

| 区域 | 默认规则策略 |
| --- | --- |
| trusted | 允许所有的数据包。 |
| home | 拒绝流入的数据包，除非与输出流量数据包相关或是 ssh,mdns,ipp-client,samba-client 与 dhcpv6-client 服务则允许。 |
| internal | 等同于 home 区域 |
| work | 拒绝流入的数据包，除非与输出流量数据包相关或是 ssh,ipp-client 与 dhcpv6-client 服务则允许。 |
| public | 拒绝流入的数据包，除非与输出流量数据包相关或是 ssh,dhcpv6-client 服务则允许。 |
| external | 拒绝流入的数据包，除非与输出流量数据包相关或是 ssh 服务则允许。 |
| dmz | 拒绝流入的数据包，除非与输出流量数据包相关或是 ssh 服务则允许。 |
| block | 拒绝流入的数据包，除非与输出流量数据包相关。 |
| drop | 拒绝流入的数据包，除非与输出流量数据包相关。 |

简单来讲就是为用户预先准备了几套规则集合，我们可以根据场景的不同选择合适的规矩集合，而默认区域是 **public**。

7.3.2 字符管理工具

如果想要更高效的配置妥当防火墙，那么就一定要学习字符管理工具 firewall-cmd 命令,命令参数有：

| 参数 | 作用 |
| --- | --- |
| --get-default-zone | 查询默认的区域名称。 |
| --set-default-zone=<区域名称> | 设置默认的区域，永久生效。 |
| --get-zones | 显示可用的区域。 |
| --get-services | 显示预先定义的服务。 |
| --get-active-zones | 显示当前正在使用的区域与网卡名称。 |
| --add-source= | 将来源于此 IP 或子网的流量导向指定的区域。 |
| --remove-source= | 不再将此 IP 或子网的流量导向某个指定区域。 |
| --add-interface=<网卡名称> | 将来自于该网卡的所有流量都导向某个指定区域。 |
| --change-interface=<网卡名称> | 将某个网卡与区域做关联。 |
| --list-all | 显示当前区域的网卡配置参数，资源，端口以及服务等信息。 |
| --list-all-zones | 显示所有区域的网卡配置参数，资源，端口以及服务等信息。 |

| --add-service=<服务名> | 设置默认区域允许该服务的流量。 |
| --- | --- |
| --add-port=<端口号/协议> | 允许默认区域允许该端口的流量。 |
| --remove-service=<服务名> | 设置默认区域不再允许该服务的流量。 |
| --remove-port=<端口号/协议> | 允许默认区域不再允许该端口的流量。 |
| --reload | 让“永久生效”的配置规则立即生效，覆盖当前的。 |

特别需要注意的是firewalld服务有两份规则策略配置记录，必需要能够区分：

RunTime:当前正在生效的。

Permanent:永久生效的。

当下面实验修改的是永久生效的策略记录时，必须执行” – reload“参数后才能立即生效，否则要重启后再生效。

**查看当前的区域：**

[root@linuxprobe ~]# firewall-cmd --get-default-zone
public

**查询eno16777728网卡的区域：**

[root@linuxprobe ~]# firewall-cmd --get-zone-of-interface=eno16777728
public

**在public中分别查询ssh与http服务是否被允许：**

[root@linuxprobe ~]# firewall-cmd --zone=public --query-service=ssh
yes
[root@linuxprobe ~]# firewall-cmd --zone=public --query-service=http
no

**设置默认规则为dmz：**

[root@linuxprobe ~]# firewall-cmd --set-default-zone=dmz

**让“永久生效”的配置文件立即生效：**

[root@linuxprobe ~]# firewall-cmd --reload
success

**启动/关闭应急状况模式，阻断所有网络连接：**

应急状况模式启动后会禁止所有的网络连接，一切服务的请求也都会被拒绝，当心，请慎用。

[root@linuxprobe ~]# firewall-cmd --panic-on
success
[root@linuxprobe ~]# firewall-cmd --panic-off
success

如果您已经能够完全理解上面练习中firewall-cmd命令的参数作用，不妨来尝试完成下面的模拟训练吧：

**模拟训练A:允许https服务流量通过public区域，要求立即生效且永久有效：**

方法一:分别设置当前生效与永久有效的规则记录：

[root@linuxprobe ~]# firewall-cmd --zone=public --add-service=https
[root@linuxprobe ~]# firewall-cmd --permanent --zone=public --add-service=https

方法二:设置永久生效的规则记录后读取记录：

[root@linuxprobe ~]# firewall-cmd --permanent --zone=public --add-service=https

[root@linuxprobe ~]# firewall-cmd --reload

**模拟训练 B:不再允许 http 服务流量通过 public 区域，要求立即生效且永久生效：**

[root@linuxprobe ~]# firewall-cmd --permanent --zone=public --remove-service=http
success

使用参数"--reload"让永久生效的配置文件立即生效：

[root@linuxprobe ~]# firewall-cmd --reload
success

**模拟训练 C:允许 8080 与 8081 端口流量通过 public 区域，立即生效且永久生效：**

[root@linuxprobe ~]# firewall-cmd --permanent --zone=public --add-port=8080-8081/tcp
[root@linuxprobe ~]# firewall-cmd --reload

**模拟训练 D:查看模拟实验 C 中要求加入的端口操作是否成功：**

[root@linuxprobe ~]# firewall-cmd --zone=public --list-ports
8080-8081/tcp
[root@linuxprobe ~]# firewall-cmd --permanent --zone=public --list-ports
8080-8081/tcp

**模拟实验 E:将 eno16777728 网卡的区域修改为 external，重启后生效：**

[root@linuxprobe ~]# firewall-cmd --permanent --zone=external --change-interface=eno16777728
success
[root@linuxprobe ~]# firewall-cmd --get-zone-of-interface=eno16777728
public

**再次提示:**请读者们再仔细琢磨下**立即生效**与**重启后依然生效**的差别，千万不要修改错了。

**模拟实验 F:设置富规则，拒绝 192.168.10.0/24 网段的用户访问 ssh 服务：**

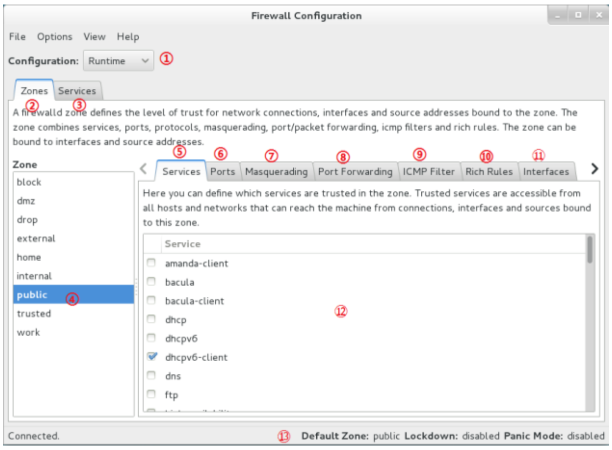
firewalld 服务的**富规则**用于对服务、端口、协议进行更详细的配置，规则的优先级最高。

[root@linuxprobe ~]# firewall-cmd --permanent --zone=public --add-rich-rule="rule family="ipv4" source address="192.168.10.0/24" service name="ssh" reject" success

7.3.3 图形管理工具

执行 **firewall-config** 命令即可看到 firewalld 的防火墙图形化管理工具，真的很强大，可以完成很多复杂的工作。

**firewalld 防火墙图形化管理工具界面详解：**




①:选择”立即生效“或”重启后依然生效“配置。

②:区域列表。

③:服务列表。

④:当前选中的区域。

⑤:被选中区域的服务。

⑥:被选中区域的端口。

⑦:被选中区域的伪装。

⑧:被选中区域的端口转发。

⑨:被选中区域的 ICMP 包。

⑩:被选中区域的富规则。

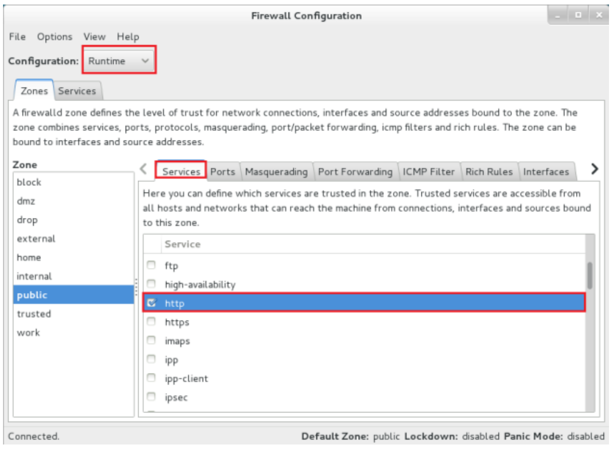
⑪被选中区域的网卡设备。

⑫被选中区域的服务，前面有√的表示允许。

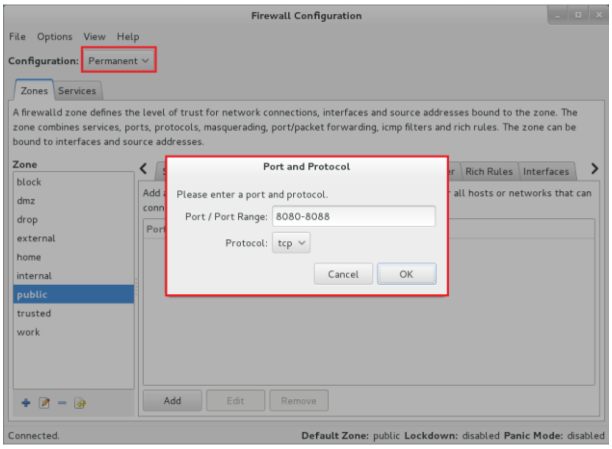
⑬firewalld 防火墙的状态

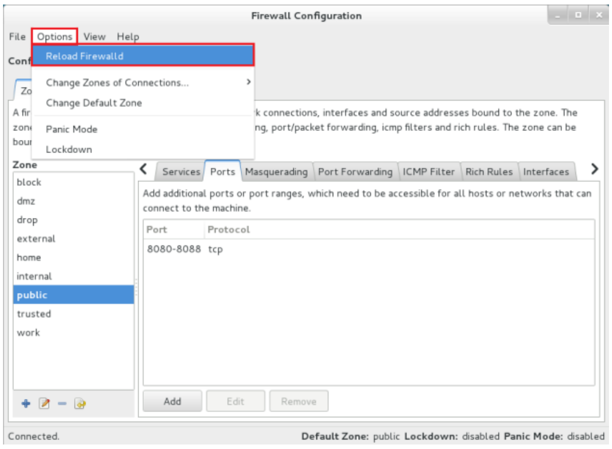
**请注意:firewall-config 图形化管理工具中没有保存/完成按钮，只要修改就会生效。**

**允许其他主机访问 http 服务，仅当前生效：**



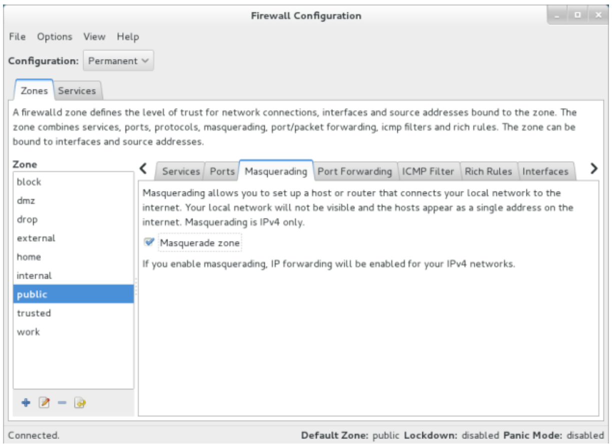
允许其他主机访问 8080-8088 端口且重启后依然生效：



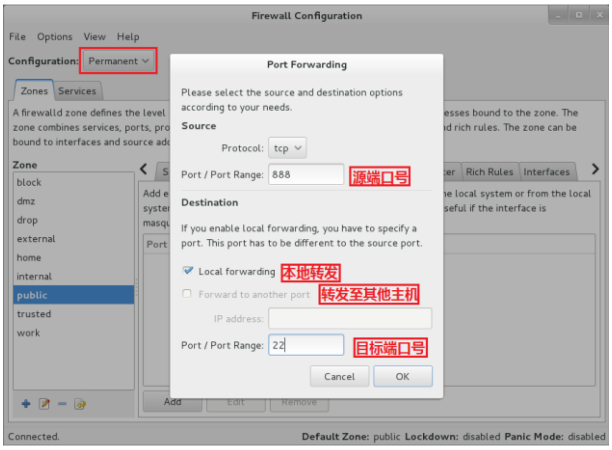




开启伪装功能，重启后依然生效：

firewalld 防火墙的伪装功能实际就是 SNAT 技术，即让内网用户不必在公网中暴露自己的真实 IP 地址。

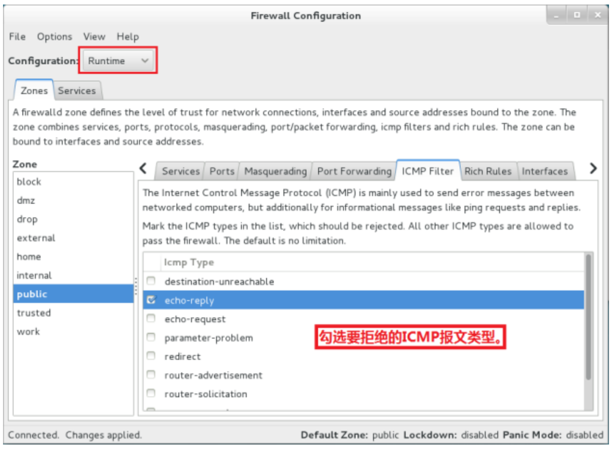



将向本机 888 端口的请求转发至本机的 22 端口且重启后依然生效：



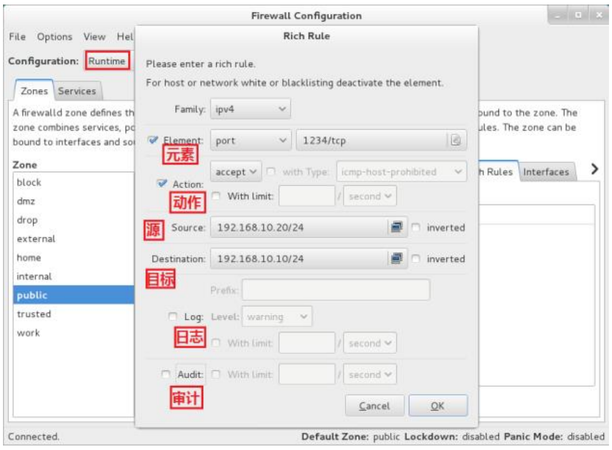
**过滤所有”echo-reply”的 ICMP 协议报文数据包，仅当前生效：**

ICMP 即互联网控制报文协议”Internet Control Message Protocol “，归属于 TCP/IP 协议族，主要用于检测网络间是否可通信、主机是否可达、路由是否可用等网络状态，并不用于传输用户数据。

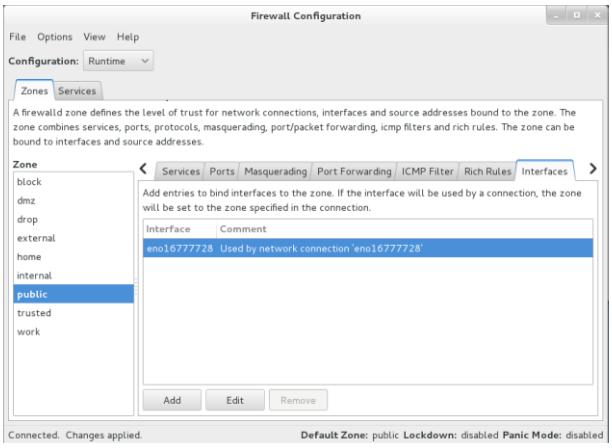



**仅允许 192.168.10.20 主机访问本机的 1234 端口，仅当前生效：**

富规则代表着更细致、更详细的规则策略，针对某个服务、主机地址、端口号等选项的规则策略，**优先级最高。**




**查看网卡设备信息：**




firewall-config 图形管理工具真的非常实用，很多原本复杂的长命令被用图形化按钮替代，设置规则也变得简单了，日常工作中真的非常实用。所以有必要跟们读者讲清配置防火墙的原则——只要能实现需求的功能，无论用文本管理工具还是图形管理工具都是可以的。

7.4 服务的访问控制列表

Tcp_wrappers(即 Transmission Control Protocol(TCP)Wrappers)是一款基于 IP 层的 ACL 访问控制列表流量监控程序，它能够根据来访主机地址与本机目标服务程序做允许或拒绝规则，控制列表修改后会立即生效，系统将会先检查允许规则，如果匹配允许则直接放行流量，若拒绝规则中匹配则直接拒绝，都不匹配默认也会放行。

允许名单:/etc/hosts.allow

拒绝名单:/etc/hosts.deny

**指定客户端的规则如下:**

| 客户端类型 | 示例 | 满足示例的客户端列表 |
|---|---|---|
| 单一主机 | 192.168.10.10 | IP 地址为 192.168.10.10 的主机。 |
| 指定网段 | 192.168.10. | IP 段为 192.168.10.0/24 的主机。 |
| 指定网段 | 192.168.10.0/255.255.255.0 | IP 段为 192.168.10.0/24 的主机。 |
| 指定 DNS 后缀 | .linuxprobe.com | 所有 DNS 后缀为.linuxprobe.com 的主机 |
| 指定主机名称 | boss.linuxprobe.com | 主机名称为 boss.linuxprobe.com 的主机。 |
| 指定所有客户端 | ALL | 所有主机全部包括在内。 |

**限制只有 192.168.10.0/24 网段的主机可以访问本机的 httpd 服务:**

编辑允许规则:

[root@linuxprobe ~]# vim /etc/hosts.allow

httpd:192.168.10.

拒绝其他所有的主机:

[root@linuxprobe ~]# vim /etc/hosts.deny

httpd:*

**本章结束，您可以在此写下笔记:**

# 第 8 章 使用 ssh 服务管理远程主机。

**章节简述：**

红帽 RHEL7 系统将原先熟悉的守护进程替换为了 **systemd**，用 sytemctl 命令替换掉了很多管理命令，变化的确很大。

学习使用 nmtui 命令配置网卡参数、手工将多块网卡做绑定、使用 nmcli 命令查看网卡信息和使用 ss 命令查看网络及端口状态。

完整演示 sshd 服务配置方法并详细讲述每个参数的作用，实战基于密钥远程登陆实验以及用 screen 服务让远程会话不再终断。

## 8.1 进程与服务

### 8.1.1 初始化进程

Linux 操作系统开机过程首先从 BIOS 开始→进入"Boot Loader"→加载内核→内核的初始化→**启动初始化进程**，
初始化进程作为系统第一个进程，它需要完成相关的初始化工作，为用户提供合适的工作环境。

红帽 RHEL7 系统已经替换掉了大家熟悉的初始化进程 **System V init**，并正式采用全新的初始化进程 **systemd**。初始化进程 systemd 使用了并发启动机制，所以开机速度得到了不小的提升。虽然 **systemd** 已经表现出了优势，但一直有抵制的呼声：

吐槽 1:因作者 Lennart Poettering 就职于红帽，让其他系统的粉丝很不爽。

吐槽 2:systemd 仅仅可在 linux 系统下运行，放弃了 BSD 用户。

吐槽 3:接管了诸如 syslogd、udev、cgroup 等等服务的工作，不再甘心只做初始化进程。

吐槽 4:使用 systemd 后 RHEL7 系统变化太大，参考文档又不多，为难我们啊！

不论怎么吐槽，既然红帽 RHEL7 系统选择了 **systemd**，原先的 inittab 将已经不再起作用，也没有了"运行级别"这个概念：
Linux 系统启动时要做大量的初始化工作——例如挂载文件系统和交换分区，启动各类进程服务等等操作，这些都可以看作是一个个的单元(Unit)，分析下 nfs 服务的单元配置文件吧：

```
[root@linuxprobe ~]# cat /etc/systemd/system/nfs.target.wants/nfs-lock.service
[Unit]
Description=NFS file locking service.
//表示 rpcbind 服务必须在 nfs 服务启动前已经运行。
Requires=rpcbind.service network.target
After=network.target named.service rpcbind.service
Before=remote-fs-pre.target

[Service]
Type=forking
StandardError=syslog+console
EnvironmentFile=-/etc/sysconfig/nfs
//启动 nfs 服务前需要执行的命令：
ExecStartPre=/usr/libexec/nfs-utils/scripts/nfs-lock.preconfig
//启动 nfs 服务具体的命令语法：
ExecStart=/sbin/rpc.statd $STATDARG
# Make sure lockd's ports are reset
ExecStopPost=-/sbin/sysctl -w fs.nfs.nlm_tcpport=0 fs.nfs.nlm_udpport=0
[Install]
WantedBy=nfs.target
```

如前面所述在红帽 RHEL7 系统中 **systemd** 用”**目标(target)**“代替了“**运行级别**”这个概念。

| Sysvinit 运行级别 | Systemd 目标名称 | 作用 |
| --- | --- | --- |
| 0 | runlevel0.target, poweroff.target | 关机 |
| 1 | runlevel1.target, rescue.target | 单用户模式 |
| 2 | runlevel2.target, multi-user.target | 等同于级别 3 |
| 3 | runlevel3.target, multi-user.target | 多用户的文本界面 |
| 4 | runlevel4.target, multi-user.target | 等同于级别 3 |
| 5 | runlevel5.target, graphical.target | 多用户的图形界面 |
| 6 | runlevel6.target, reboot.target | 重启 |
| emergency | emergency.target | 紧急 Shell |

将默认的运行级别修改为“多用户，无图形模式”：

[root@linuxprobe ~]# ln -sf /lib/systemd/system/multi-user.target /etc/systemd/system/default.target

将默认的运行级别修改为“图形化模式”：

[root@linuxprobe ~]# ln -sf /lib/systemd/system/graphical.target /etc/systemd/system/default.target

8.1.2 管理服务命令

对于学习过红帽 RHEL6 系统或已经习惯使用 service、chkconfig 等命令来管理系统服务的读者可能要郁闷一段时间了，因为在红帽 RHEL7 系统中管理服务的命令是”**systemctl**“，但使用方法大致相同，我们来做下对比吧。

**systemctl 管理服务的启动、重启、停止、重载、查看状态的命令：**

| Sysvinit 命令(红帽 RHEL6 系统) | Systemctl 命令（红帽 RHEL7 系统） | 作用 |
| --- | --- | --- |
| service foo start | systemctl start foo.service | 启动服务 |
| service foo restart | systemctl restart foo.service | 重启服务 |
| service foo stop | systemctl stop foo.service | 停止服务 |
| service foo reload | systemctl reload foo.service | 重新加载配置文件（不终止服务） |
| service foo status | systemctl status foo.service | 查看服务状态 |

**systemctl 设置服务的开机启动、不启动、查看各级别下服务启动状态的命令：**

| Sysvinit 命令(红帽 RHEL6 系统) | Systemctl 命令（红帽 RHEL7 系统） | 作用 |
|---|---|---|
| chkconfig foo on | systemctl enable foo.service | 开机自动启动 |
| chkconfig foo off | systemctl disable foo.service | 开机不自动启动 |
| chkconfig foo | systemctl is-enabled foo.service | 查看特定服务是否为开机自启动 |
| chkconfig --list | systemctl list-unit-files --type=service | 查看各个级别下服务的启动与禁用情况 |

8.1.3 监视资源与管理进程

Linux 系统中时刻运行着许许多多的进程，如果能够合理的管理它们，绝对有益于系统的性能优化，系统进程总共有 5 种不同的状态：

- **R(运行):**正在运行或在运行队列中等待。
- **S(中断):**休眠中, 在等待某个条件的形成或接受到信号。
- **D(不可中断):**收到信号不唤醒和不可运行, 进程必须等待直到有中断发生。
- **Z:(僵死):**进程已终止, 但进程描述符存在, 直到父进程调用 wait4()系统调用后释放。
- **T:(停止):**进程收到 SIGSTOP, SIGSTP, SIGTIN, SIGTOU 信号后停止运行。

**ps 命令用于查看系统中的进程状态，格式为：“ps [参数]”。**

查看进程与状态: “ps -aux”

查找某个特定的进程信息:” ps -aux | grep 进程名”

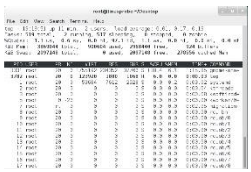
| USER | PID | %CPU | %MEM | VSZ | RSS | TTY | STAT | START | TIME | COMMAND |
|---|---|---|---|---|---|---|---|---|---|---|
| 进程的所有者 | 进程 ID 号 | 运算器占用率 | 内容占用率 | 虚拟内存使用量(单位是 KB) | 占用的固定内存量(单位是 KB) | 所在终端 | 进程状态 | 被启动的时间 | 实际使用 CPU 的时间 | 命令名称与参数 |

| 参数 | 作用 |
|---|---|
| -a | 显示所有的进程（包括其他用户的） |
| -u | 用户以及其他详细信息 |
| -x | 显示没有控制终端的进程 |

**top 命令用于监视进程的活动与系统负载，格式为：“top”。**

这个 top 命令可真的是太厉害了，完全可以比喻成是“强化版的 Windows 任务管理器”，运行界面见右图：

**前面的五行为系统整体的统计信息，下面我们来逐行的讲解：**



第 1 行:系统时间，运行时间，登陆用户数，系统负载（分别为 1 分钟、5 分钟、15 分钟的平均值）。

第 2 行:进程总数，运行中的，睡眠中的，停止的，僵尸的。

第 3 行:用户占用资源，系统内核占用资源，改变过优先级的进程，空闲的资源，等待输入输出的时间。此行数据均为 CPU 数据并以百分比格式显示，例如”99.2 id”意味着有 99.2%的 CPU 资源正在空闲中。

第 4 行:物理内存总量，使用量，空闲量，作为内核缓存的内存量。

第 5 行:虚拟内存总量，使用量，空闲量，已被提前加载的内存数据。

**进程的信息区中包含了各个进程的详细信息，含义如下：**

PID:进程 ID 号
USER:进程的所有者
PR:优先级
NI:优先级（负值表示优先级更高）
VIRT:虚拟内存使用量
RES:物理内存使用量
SHR:共享内存大小
S:进程状态（上文中有提到）
%CPU:运算器的使用百分比
%MEM:内存的使用百分比
TIME+:使用 CPU 的时间(单位是 1/100 秒)
COMMAND:命令名称

**pidof 命令用于查询某个特定程序的进程 PID 值，格式为："pidof [参数] [程序名称]"。**

查询"sshd"进程的 PID 值:"pidof sshd"

**kill 命令用于终止某个特定 PID 号码的进程，格式为："kill [参数] [进程 PID 号]"。**

强制终止 PID 为 4674 的进程:"kill -9 4674"

✪其中的"-9"代表强制终止(SIGKILL)，也是最常用的一种信号参数，查看全部请执行"kill -l"

**killall 命令用于终止某个特定名称的所有进程，格式为："killall [参数] [进程名称]"。**

终止名称为"sshd"的进程:"killall sshd"

在终端中运行一个命令后如果想立即的停止它，可以使用组合键"**Ctrl+c**"，这样命令的进程将会彻底的被终止。

但还有一种玩法是"**Ctrl+z**"，它是将命令的进程暂停（也叫挂载到后台或扔到后台)，先来看两条命令吧:

这条命令会每秒向家目录中的 jobs.txt 中追加一个字符串：

[root@linuxprobe ~]# (while true ;do echo -n " working " >> ~/jobs.txt;sleep 1 ;done;)

自动刷新查看文件内容的变化：

[root@linuxprobe ~]# tail -f ~/jobs.txt

**模拟训练:试试"Ctrl+z"，学习 jobs、bg 和 fg 命令。**

开始执行写入命令：

[root@linuxprobe ~]# (while true ;do echo -n " working " >> ~/jobs.txt;sleep 1 ;done;)

确认一直在被写入字符：

[root@linuxprobe ~]# tail -f ~/jobs.txt

敲击"**ctrl+z**"后，这条命令的进程被暂停了(另外的终端中不再被追加内容)：

[root@linuxprobe ~]# (while true ;do echo -n " working " >> ~/jobs.txt;sleep 1 ;done;)
^Z
[1]+ Stopped ( while true; do
echo -n " working " >> ~/jobs.txt; sleep 1;
done )

使用 jobs 命令可以查看到所有在后台运行着的进程:

[root@linuxprobe ~]# jobs
[1]+ Stopped ( while true; do
echo -n " working " >> ~/jobs.txt; sleep 1;
done )

运行 **bg 命令**让后台的程序继续执行，现在后台中只有一个进程，所以省略了编号，完整格式应为"**bg 1** "：

[root@linuxprobe ~]# bg

[1]+ ( while true; do
echo -n " working " >> ~/jobs.txt; sleep 1;
done )

运行 **fg 命令**将后台的进程再调回前台，程序依然在运行，此时你可以敲击组合键”**ctrl+c** “啦：

[root@linuxprobe ~]# fg
( while true; do
echo -n " working " >> ~/jobs.txt; sleep 1;
done )&

有些命令在执行时会不断的在终端上输出信息，影响到我们继续输入命令了，此时便可以在这条命令后面添加个” **&** “符号,那么从一开始执行该命令就会是在后台执行（不是在后台暂停，而是在运行的）。

8.2 配置网卡连接网络

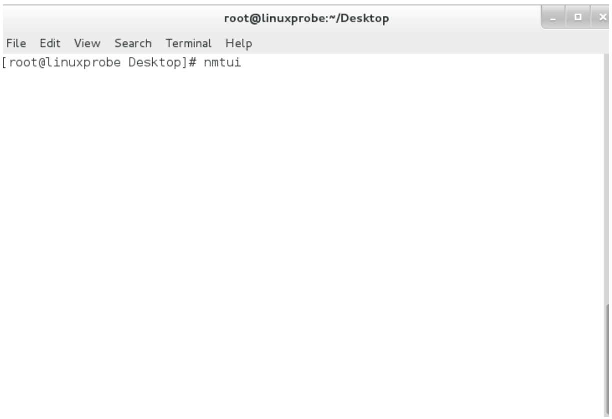
8.2.1 配置网卡参数

**本实验需要两台虚拟主机来完成，分别是：**

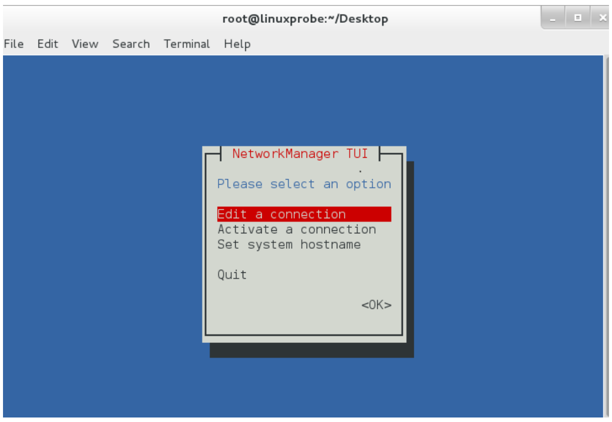
| 主机名称 | 操作系统 | IP 地址 |
| --- | --- | --- |
| 本地主机 | 红帽 RHEL7 操作系统 | 192.168.10.10 |
| 远程主机 | 红帽 RHEL7 操作系统 | 192.168.10.20 |

在正式配置 sshd 服务之前，我们必须保证本地主机与远程主机之间数据是可以互相传送的，前面在学习 Vim 编辑器的章节中修改过网卡文件，本次使用图形工具来配置网络，效果是一样的。

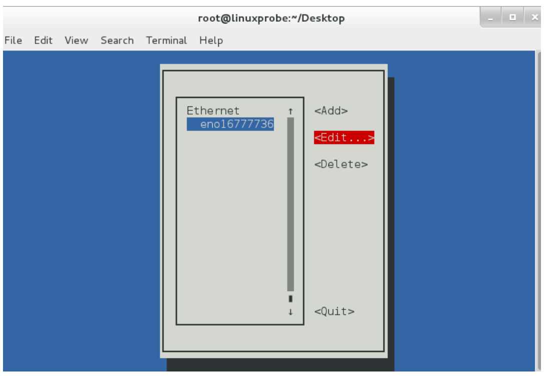
第 1 步：执行命令“nmtui”。



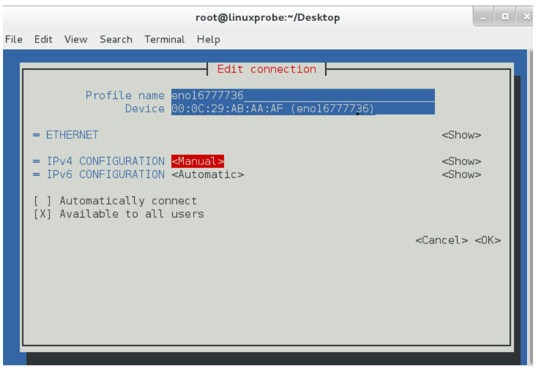

第 2 步：选择要编辑的网卡。



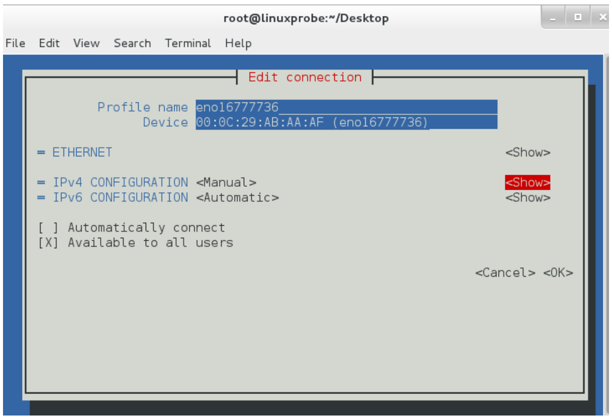

第 3 步：编辑网卡信息。



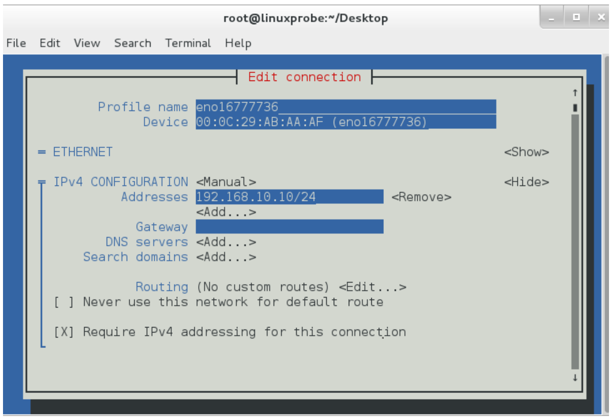

第 4 步：将网卡的 IPv4 配置项设置成手动。



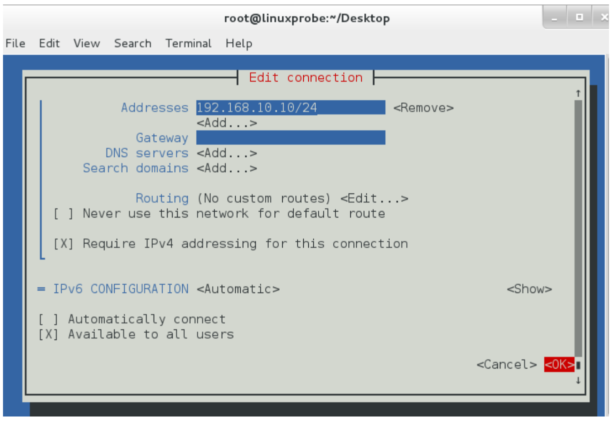
第 5 步：敲击 Ipv4 配置项右侧的(Show)。



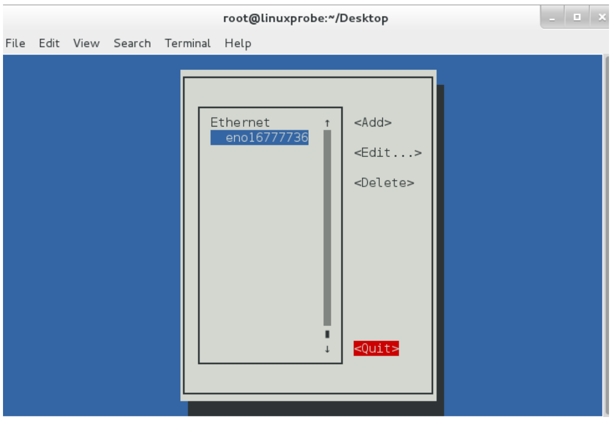

第 6 步：填入 IP 地址信息。



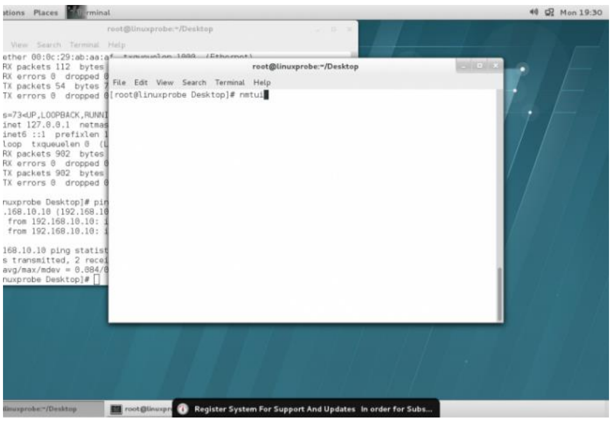

第 7 步：敲击最下面的(OK)。



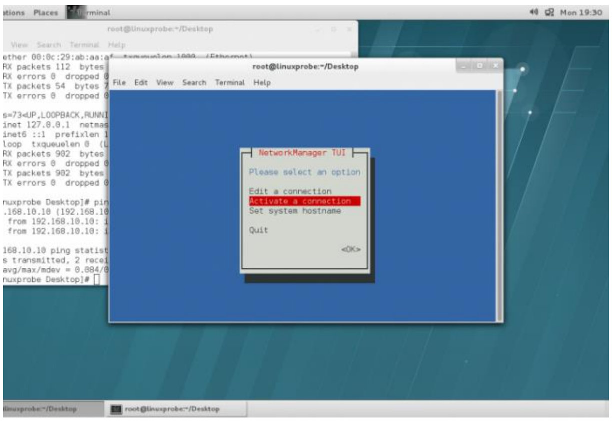

第 8 步：确认信息填写正确后退出。



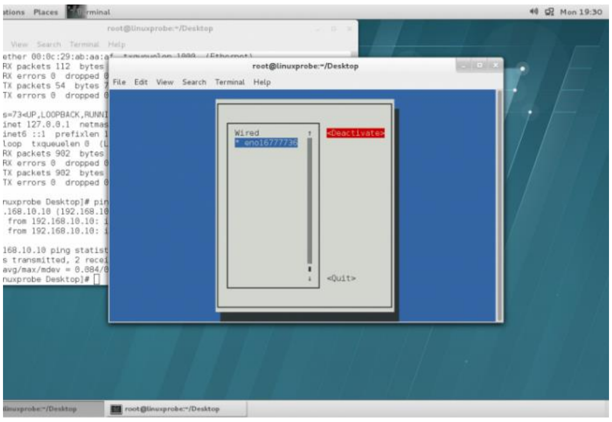

第 9 步：再次运行网卡配置程序。



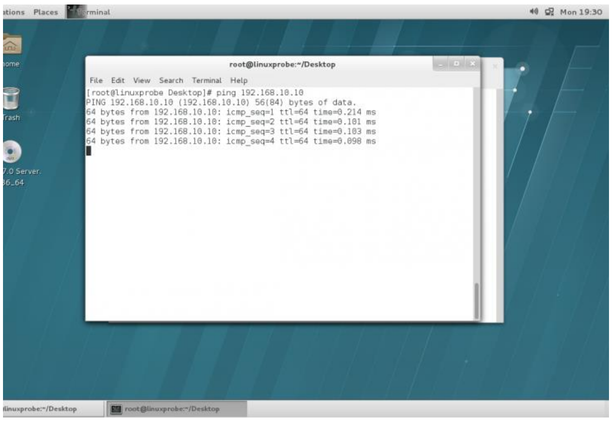

第 10 步：选择激活该网卡。



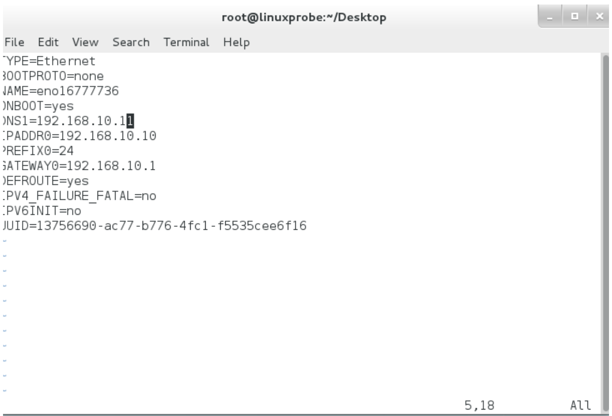
第 11 步：将此网卡成功的激活。



第 12 步：通信测试。



第 13 步：请使用"nmtui"工具配置后仍然需要修改网卡配置文件设置"ONBOOT=yes"。




当您按照上述步骤配置完网卡后在本地主机执行”ping 192.168.10.20“来检测网络是否已经配置妥当。

8.2.2 查看网卡信息

nmcli 是一款能够方便我们配置网络的工具，能够轻松的查看网卡信息或网络状态：

查看网卡的配置信息：

[root@linuxprobe ~]# **nmcli connection show**

| NAME（网卡名称） | UUID（唯一识别码） | TYPE（网卡类型） | DEVICE（设备） |
|---|---|---|---|
| eno16777736 | 13756690-ac77-b776-4fc1-f5535cee6f16 | 802-3-ethernet | eno16777736 |

查看网卡的连接状态：

[root@linuxprobe ~]# **nmcli device status**

| DEVICE（设备） | TYPE（类型） | STATE（状态） | CONNECTION（连接） |
|---|---|---|---|
| eno16777736 | ethernet | connected | eno16777736 |
| lo | loopback | unmanaged | -- |

如果想看网卡设备”eno16777736“的详细信息，只需执行”nmcli con show eno16777736“，信息相当详细哦！

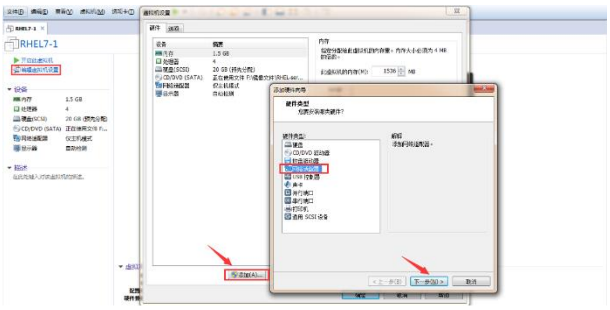
对了!网卡还支持了简单实用的多会话功能了呢，例如将 Linux 系统安装到了笔记本上，上午拿到公司工作时是要指定 IP 地址，而晚上回到家是 DHCP 分配 IP 地址，这样改来改去真的很麻烦，所以我们可以设置多个网卡会话，在不同的环境激活就可以了，**但每个网卡同时仅能有一个会话是激活状态的**。

我们可以将在公司的会话叫做”company“,在家里的会话叫做”house“，记住了哦，现在配置！
添加公司会话，参数为 connection(会话),add(添加动作),con-name(会话名称),type(网卡类型),ifname(网卡名称)：
[root@linuxprobe ~]# *nmcli connection add con-name company type ethernet ifname eno16777736*
Connection 'company' (3a6677a8-59b0-4c8a-ae15-2a9f3e502f33) successfully added.
添加居家会话：
[root@linuxprobe ~]# *nmcli connection add con-name house ifname eno16777736 autoconnect no type ethernet ip4 192.168.10.10/24 gw4 192.168.10.1*
Connection 'house' (03f366a3-04b6-4545-a996-f10d7bffbf64) successfully added.
启用居家会话：
[root@linuxprobe ~]# nmcli connection up house
**查看会话的信息：**
[root@linuxprobe ~]# nmcli connection show

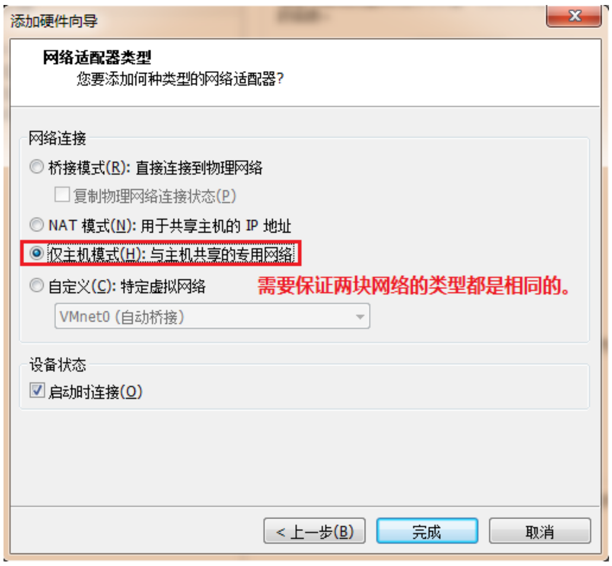
| NAME（名称） | UUID（唯一标识符） | TYPE （网卡类型） | DEVICE（设备） |
|---|---|---|---|
| house | 03f366a3-04b6-4545-a996-f10d7bffbf64 | 802-3-ethernet | -- |
| company | 3a6677a8-59b0-4c8a-ae15-2a9f3e502f33 | 802-3-ethernet | -- |
| eno16777736 | 13756690-ac77-b776-4fc1-f5535cee6f16 | 802-3-ethernet | eno16777736 |

实用 nmcli 命令创建或修改的会话配置信息回自动保存为网卡配置文件，重启后依然有效。

8.2.3 绑定两块网卡
我们可以将多块网卡多绑定操作，不仅能够提高带宽的速率而且让其中一块网卡出现故障时，不会让网络完全中断。
**第 1 步:在虚拟机中额外添加一块网卡。**
**编辑虚拟机设置，添加网络适配器：**




**保证两块网卡的连接类型都是相同的：**




查看两块网卡的名称：
[root@linuxprobe ~]# ifconfig | grep flags
eno16777728: flags=4163>UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
eno33554968: flags=4163>UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

**第 2 步:配置网卡的参数:**

设置第 1 块网卡为从卡，而主卡为 bond0:

```
[root@linuxprobe ~]# vim /etc/sysconfig/network-scripts/ifcfg-eno16777728
  TYPE=Ethernet
  BOOTPROTO=none
  ONBOOT=yes
  USERCTL=no
  DEVICE=eno16777728
  MASTER=bond0
  SLAVE=yes
```

相似的方法设置第 2 块网卡，主卡依然为 bond0:

```
[root@linuxprobe ~]# vim /etc/sysconfig/network-scripts/ifcfg-eno33554968
  TYPE=Ethernet
  BOOTPROTO=none
  ONBOOT=yes
  USERCTL=no
  DEVICE=eno33554968
  MASTER=bond0
  SLAVE=yes
```

创建绑定网卡的配置文件并指定 IP 地址等信息:

```
[root@linuxprobe ~]# vim /etc/sysconfig/network-scripts/ifcfg-bond0
  TYPE=Ethernet
  BOOTPROTO=none
  ONBOOT=yes
  USERCTL=no
  DEVICE=bond0
  IPADDR=192.168.10.10
  PREFIX=24
  DNS=192.168.10.1
  NM_CONTROLLED=no
```

**第 3 步:让内核支持 Bonding 的驱动。**

为 bond0 网卡添加 bonding 驱动的支持:

```
[root@linuxprobe ~]# vim /etc/modprobe.d/bond.conf
alias bond0 bonding
options bond0 miimon=100 mode=6
```

常用的绑定驱动模式有:

mode=0 平衡负载模式:平时两块网卡均工作，且自动备援，采用 Switch 支援。

mode=1 自动备援模式:平时只有一块网卡工作，故障后自动替换为另外的网卡。

mode=6:平衡负载模式:平时两块网卡均工作，且自动备援，无须设置 Switch 支援。

**第 4 步:重新加载网卡后绑定即成功。**

重新加载网卡信息:

```
[root@linuxprobe ~]# systemctl restart network
```

8.2.4 查看端口状态

有经验的管理员都会在配置网卡后顺手执行一条 ping 命令来检测网络的可用性，并且以前大家习惯用 netstat 命令查看本机的端口连接状态，这条命令也已经在红帽 RHEL7 系统中则被效率更高、显示信息更多的 ss 命令替代了。

**ss 命令用于查看本机的端口连接状态，具体的参数:**

| 参数 | 作用 |
| --- | --- |
| -a | 显示所有的套接字 |
| -l | 显示所有连接状态的套接字 |
| -e | 显示详细的套接字信息 |
| -m | 显示套接字的内存使用情况 |
| -p | 显示套接字的进程信息 |
| -4 | 显示 ipv4 的套接字信息 |
| -6 | 显示 ipv6 的套接字信息 |
| -t | 仅显示 tcp 的套接字信息 |
| -u | 仅显示 udp 的套接字信息 |
| -n | 不解析主机名（提升速度） |
| -s | 查看概述 |

查看监听状态中的套接字:

[root@linuxprobe ~]# **ss -ntl**

| State | Recv-Q | Send-Q | Local Address:Port | Peer Address:Port |
| --- | --- | --- | --- | --- |
| LISTEN | 0 | 100 | 127.0.0.1:25 | *:* |
| LISTEN | 0 | 128 | *:55820 | *:* |
| LISTEN | 0 | 100 | *:22 | *:* |
| LISTEN | 0 | 128 | 127.0.0.1:631 | *:* |
| LISTEN | 0 | 128 | :::60863 | *:* |

查看进程名和 PID 号码：
[root@linuxprobe ~]# ss -s

| Total: 1091 (kernel 1173) | | | |
|---|---|---|---|
| TCP: 11 (estab 0, closed 1, orphaned 0, synrecv 0, timewait 0/0), ports 0 | | | |
| Transport | Total | IP | IPv6 |
| * | 1173 | - | - |
| RAW | 0 | 0 | 0 |
| UDP | 13 | 8 | 5 |
| TCP | 10 | 5 | 5 |
| INET | 23 | 13 | 10 |
| FRAG | 0 | 0 | 0 |

如果我们希望查看 IP 数据包从本机到另外一台电脑经过的路由信息，那就可以用 **tracepath** 命令啦！
这里格式非常简单:"tracepath 目标地址(域名或 IP 地址均可)"，确实没什么可讲。
[root@linuxprobe ~]# **tracepath www.linuxprobe.com**
追踪从本地主机到《Linux 就该这么学》的服务器中数据包经过了那些路由器，般来讲路由跳数越少，延时越低，访问速度越快哦！

8.3 远程控制服务
8.3.1 了解 sshd 服务
SSH(Secure Shell)是一种能够提供安全远程登陆会话的协议，假如希望在远程 Linux 系统中执行命令，就是通过这个协议啦！
为什么要强调 SSH 协议是安全的呢？因为比如 ftp、telnet 等服务在网络上不会对口令或数据进行加密，那么骇客们真的非常容易就可以截获这些信息（尤其是同局域网内的用户），因此它们在本质是就是不安全的。
**sshd 服务提供两种安全验证的方法：**
**基于口令的安全验证:**经过验证帐号与密码即可登陆到远程主机。
**基于密钥的安全验证:**需要在本地生成"**密钥对**"后将公钥传送至服务端，进行公共密钥的比较。
**sshd 服务的配置文件解析：**
[root@linuxprobe ~]# cat /etc/ssh/sshd_config

| 参数 | 作用 |
|---|---|
| #Port 22 | 默认的 sshd 服务端口。 |
| #ListenAddress 0.0.0.0 | 设定 sshd 服务端监听的 IP 地址。 |
| #Protocol 2 | SSH 协议的版本号。 |
| #HostKey /etc/ssh/ssh_host_key | SSH 协议版本为 1 时，私钥存放的位置。 |

| | |
|---|---|
| HostKey /etc/ssh/ssh_host_rsa_key | SSH 协议版本为 2 时，RSA 私钥存放的位置。 |
| #HostKey /etc/ssh/ssh_host_dsa_key | SSH 协议版本为 2 时，DSA 私钥存放的位置。 |
| #PermitRootLogin yes | 设定是否允许 root 用户直接登录。 |
| #StrictModes yes | 当远程用户私钥改变时则直接拒绝连接。 |
| #MaxAuthTries 6 | 最大密码尝试次数 |
| #MaxSessions 10 | 最大终端数 |
| #PasswordAuthentication yes | 是否允许密码验证 |
| #PermitEmptyPasswords no | 是否允许空密码登陆（很不安全） |

若您想要修改服务的配置参数，请一定要记得删除参数前面的注释符”#“并重启服务才生效的。

**在远程主机上启动 sshd 服务并加入到开机启动项：**

[root@localhost ~]# systemctl start sshd
[root@localhost ~]# systemctl enable sshd

8.3.2 使用 ssh 命令

**ssh 命令用于远程管理 Linux 主机，格式为：“ssh [参数] 主机”。**

| 参数 | 作用 |
|---|---|
| -p | 指定连接端口(默认为 22) |
| -v | 显示连接过程的详细信息 |

[root@localhost ~]# **ssh 192.168.10.20**
The authenticity of host '192.168.10.20 (192.168.10.20)' can't be established.
ECDSA key fingerprint is 4f:a7:91:9e:8d:6f:b9:48:02:32:61:95:48:ed:1e:3f.
Are you sure you want to continue connecting (yes/no)? **yes**
Warning: Permanently added '192.168.10.20' (ECDSA) to the list of known hosts.
root@192.168.10.20's password:**此处输入远程主机 root 用户的密码**
Last login: Wed Apr 15 15:54:21 2015 from 192.168.10.10
[root@localhost ~]#

8.3.3 安全密钥验证

使用密码验证终归会存在着被骇客暴力破解或嗅探监听的危险，其实也可以让 ssh 服务**基于密钥进行安全验证**（可无需密码验证）。

**第 1 步:在本地主机中生成“密钥对”并将公钥传送到远程主机中：**

[root@linuxprobe ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):**回车或设置密钥的存储路径**
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase): **回车或设置密钥的密码**

Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
40:32:48:18:e4:ac:c0:c3:c1:ba:7c:6c:3a:a8:b5:22 root@linuxprobe.com
The key's randomart image is:

```
+--[ RSA 2048]----+
|+*..o .          |
|*.o   +          |
|o*      .        |
|+ .     .        |
|o..      S       |
|.. +             |
|. =              |
|E+ .             |
|+.o              |
+-----------------+
```

**将生成好的公钥密钥传送至远程主机：**

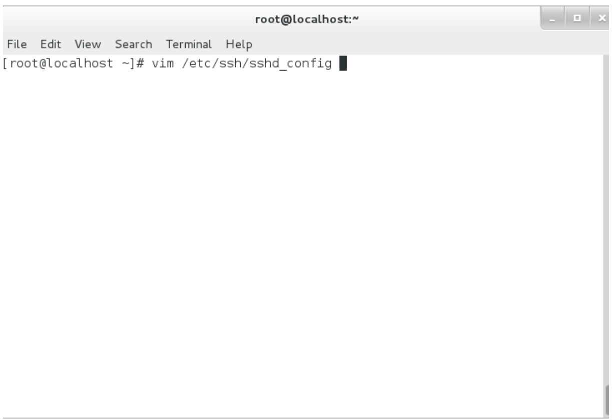
[root@linuxprobe ~]# ssh-copy-id 192.168.10.20
The authenticity of host '192.168.10.20 (192.168.10.20)' can't be established.
ECDSA key fingerprint is 4f:a7:91:9e:8d:6f:b9:48:02:32:61:95:48:ed:1e:3f.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
root@192.168.10.20's password:

Number of key(s) added: 1

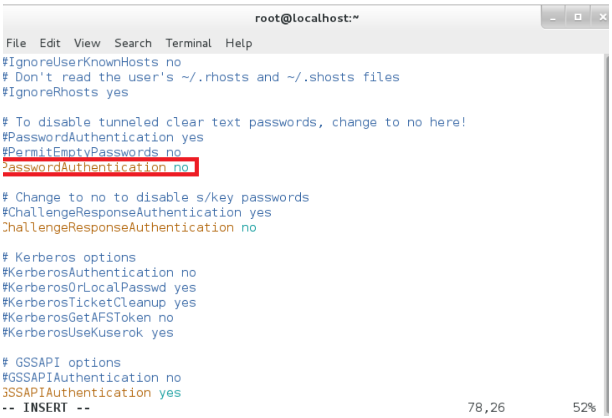
Now try logging into the machine, with: "ssh '192.168.10.20'"
and check to make sure that only the key(s) you wanted were added.

**第 2 步:首先要在远程主机中修改 sshd 服务的配置文件（修改后记得重启服务）：**

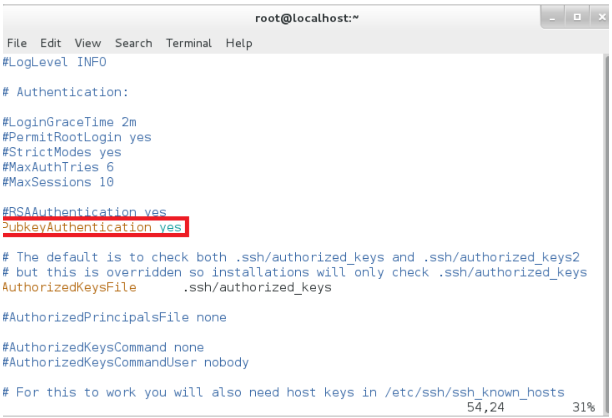
第 1 步：编辑 ssh 服务程序主配置文件。



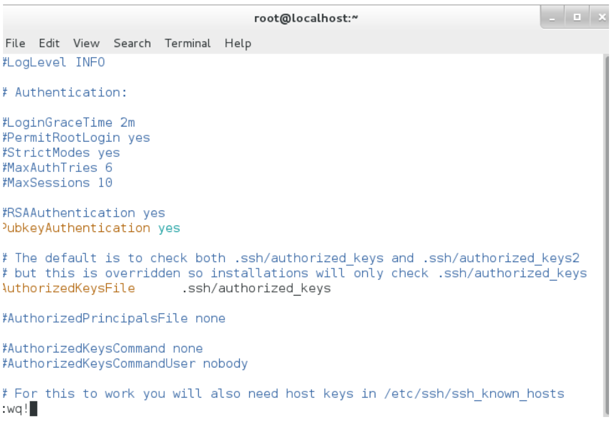
第 2 步：将允许密码验证的参数设置为 no。



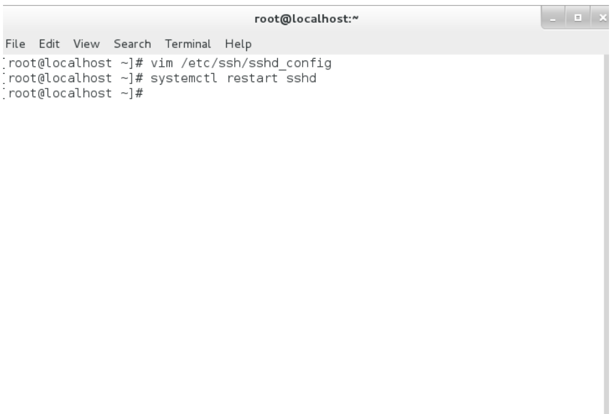

第 3 步：将允许密钥验证的参数设置为 yes。




第 4 步：保存并退出配置文件。




第 5 步：重启 ssh 服务程序后即可生效。



**第 3 步:最后来尝试无需密码的远程登录吧:**

[root@linuxprobe ~]# ssh 192.168.10.20

Last login: Mon Apr 13 19:34:13 2015

### 8.3.4 远程传输命令

要想将一些文件通过网络传送给其他主机，又恰好两台主机都是 Linux 系统，我们便可以直接用 scp 命令传输文件到另外一台主机~

**scp 命令用于在网络中安全的传输文件，格式为：“scp [参数] 本地文件 远程帐户@远程 IP 地址:远程目录”。**

| 参数 | 作用 |
| --- | --- |
| -v | 显示详细的连接进度 |
| -P | 指定远程主机的 sshd 端口号 |
| -r | 传送文件夹时请加此参数 |
| -6 | 使用 ipv6 协议 |

将本地文件/root/out.txt 传送到远程主机的/home 目录：

[root@linuxprobe ~]# scp /root/out.txt 192.168.10.20:/home

root@192.168.10.20's password:**此处输入远程主机中 root 用户的密码**

out.txt 100% 0 0.0KB/s 00:00

传送下文件夹并指定远程用户：

[root@linuxprobe ~]# scp -r results/ linuxprobe@192.168.10.20:/home

linuxprobe@192.168.10.20's password:**此处输入远程主机中 linuxprobe 用户的密码**

强大的 scp 命令还可以将远程主机的文件传输到本地呢，格式为”**scp [参数] 远程用户@远程 IP 地址:远程文件 本地目录**“。

将远程主机的/etc/issue.net 文件下载到本地的/root 目录：

[root@linuxprobe ~]# scp linuxprobe@192.168.10.20:/etc/issue.net /root

linuxprobe@192.168.10.20's password:

issue.net 100% 22 0.0KB/s 00:00

## 8.4 不间断会话服务

### 8.4.1 了解 Screen 服务

学完了 ssh 服务后有没有发现一个很重要的事情——**当连接的终端被关闭时，运行在服务器上的命令也会中断**。如果有长时间文件备份或 FTP 传输等任务时，通常我们都会新开一个连接窗口再继续工作，并且中途不能关闭窗口或断开链接（**也包括网络不稳定的情况**），否则这个任务就会被中断，还要重新开始。

Screen 便是为了解决上述问题而设计的，用户可以通过使用 Screen 命令同时控制多个命令行会话并自由切换，特点有：

**会话恢复**:即便网络中断，也可让会话随时恢复，用户不会失去对命令行的控制。

**多窗口**:每个会话都是独立运行的，拥有独立的编号、输入输出和窗口缓存。

**会话共享**:可以使多个用户从不同终端使用同一个会话，也可让他们看到完全相同的输出。

### 8.4.2 掌握命令参数

红帽 RHEL7 系统中默认没有包含 screen，需要先来安装。

**使用 yum 命令安装 screen 程序包：**

[root@linuxprobe ~]# yum install screen
Loaded plugins: langpacks, product-id, subscription-manager
………………省略部分安装信息………………
Installing:
screen x86_64 4.1.0-0.19.20120314git3c2946.el7 rhel7 551 k
………………省略部分安装信息………………
Complete!

**screen 命令的常用命令参数包括：**

| 参数 | 作用 |
| --- | --- |
| -A | 让所有视窗自动调整适应当前终端机的大小。 |
| -d <会话名称> | 将指定的 screen 会话离线。 |
| -r<会话名称> | 将指定的 screen 会话恢复。 |
| -h<行数> | 指定视窗的缓冲区行数。 |
| -S<会话名称> | 指定 screen 会话的名称 |
| -x | 恢复所有离线的会话。 |
| -ls 或-list | 显示当前的 screen 会话。 |
| --wipe | 自动将无法使用的 screen 会话删除。 |

### 8.4.3 创建与使用会话功能

完成安装后直接运行 screen 即可使用服务，推荐为每个会话都取一个名字，方便分辨。

创建名称为 backup 的会话：

[root@linuxprobe ~]# screen -S backup

查看当前已经存在的会话：

[root@linuxprobe ~]# screen -ls
There is a screen on:
32230.backup (Attached)
1 Socket in /var/run/screen/S-root.

当执行 screen 命令后会调用系统默认的 shell(通常即 bash)，所以敲完 screen 命令后会立即返回一个命令提示符，虽然看起来与刚刚没有变化，**但此时你已经进入 screen 会话啦！**

创建一个会话，初始为用 vim 编辑器写文件：

[root@linuxprobe ~]# screen vim memo.txt

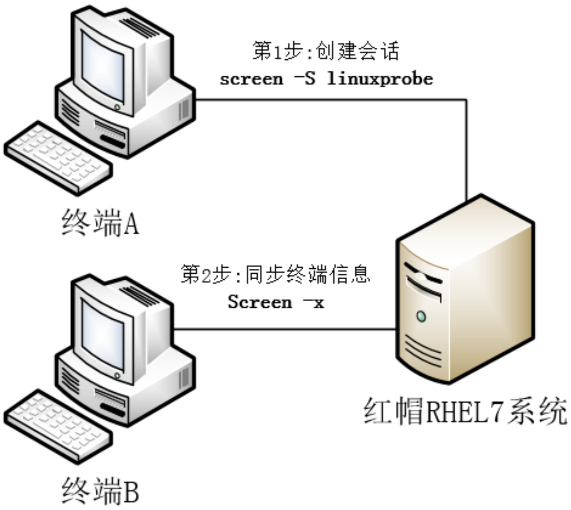
退出 vim 后会话也会被自动被删除：

[root@linuxprobe ~]# screen -ls
There is a screen on:

32230.backup (Attached)
1 Socket in /var/run/screen/S-root.
新建一个叫做 linuxprobe 的会话：
[root@linuxprobe ~]# screen -S linuxprobe
列出当前所有会话（有两个哦）：
[root@linuxprobe ~]# screen --ls
There are screens on:
32403.linuxprobe (Attached)
32230.backup (Attached)
2 Sockets in /var/run/screen/S-root.
回到 backup 会话中：
[root@linuxprobe ~]# screen -r backup
将 linuxprobe 会话离线：
[root@linuxprobe ~]# screen -d linuxprobe
[remote detached from 32403.linuxprobe]
再次查看会话状态(linuxprobe 已经被离线了):
[root@linuxprobe ~]# screen -ls
There are screens on:
32403.linuxprobe (Detached)
32230.backup (Attached)
2 Sockets in /var/run/screen/S-root.
将当前会话离线并回到 linuxprobe 会话中：
[root@linuxprobe ~]# screen -d -r linuxprobe
[32403.linuxprobe detached.]
回到 linuxprobe 会话后，状态又改变了：
[root@linuxprobe ~]# screen -ls
There are screens on:
32403.linuxprobe (Attached)
32230.backup (Attached)
2 Sockets in /var/run/screen/S-root.
[/pre]
总结来说：将 screen 会话甚至为暂时断开(**detach**)，那么会话窗口中的程序依然会执行。随后将会话重新连接(**attach**)，那么即可重新控制会话窗口中运行的程序啦。

**8.4.4 会话共享功能**

会话共享功能是一件很酷的事，它让多个用户同时使用某一个会话，甚至让您和对方看到相同的终端内容，拓扑如下：




**将两台 Linux 主机均连入同一个服务器:**

在主机 A 的终端上执行创建会话的操作:

[root@linuxprobe ~]# screen -S linuxprobe

在主机 B 的终端上同步会话信息:

[root@linuxprobe ~]# screen -x

那么此时终端 A 与终端 B 上做的任何操作，都可以实时同步到对方的屏幕上，真的很酷哦!

第 9 章 使用 Apache 服务部署静态网站。

**章节简述：**

本章节中通过对比目前热门的网站服务程序来说明 Apache 服务程序的优势，并新增主机空间选购技巧小节。

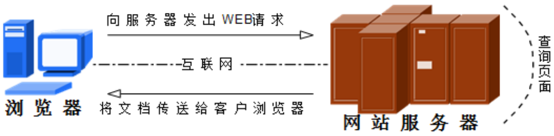
了解 SELinux 服务的 3 种工作模式，小心谨慎的使用 semanage 命令和 setsebool 命令配置 SELinux 安全上下文和服务监管策略。

学习 Apache 网站服务程序的基本部署、个人用户主页功能以及基于 IP 地址、主机名（域名）、端口号的虚拟主机功能。

### 9.1 网站服务程序

Web 网络服务也叫 WWW(World Wide Web)，一般是指能够让用户通过浏览器访问到互联网中文档等资源的服务。

目前提供 WEB 网络服务的程序有 Apache、Nginx 或 IIS 等等，Web 网站服务是被动程序，即只有接收到互联网中其他计算机发出的请求后才会响应，然后 Web 服务器才会使用 HTTP(超文本传输协议)或 HTTPS（超文本安全传输协议）将指定文件传送到客户机的浏览器上。




现在既然知道了 Web 网站服务的原理，那么都有那些程序可以提供 Web 服务那？又各自有何优势？来一起分析下吧：




Windows 系统中默认 Web 服务程序是 IIS(Internet Information Services)，这是一款图形化的网站管理工具，IIS 程序不光能提供 Web 网站服务，还能够提供 FTP、NMTP、SMTP 等服务功能，**但只能在 Windows 系统中使用**。

nginx——最初于 2004 年 10 月 4 日为俄罗斯知名门户站点而开发的，作为一款轻量级的网站服务软件，因其稳定性和丰富的功能而深受信赖，但最最最被认可的是**低系统资源**、**占用内存少**且**并发能力强**，目前国内如新浪、网易、腾讯等门户站均使用。

**Apache**——取自美国印第安人土著语 Apache,寓意着拥有高超的作战策略和无穷的耐性，由于其跨平台和安全性广泛被认可且拥有快速、可靠、简单的 API 扩展。目前拥有很高的 Web 服务软件市场占用率，全球使用最多的 Web 服务软件，开源、跨平台（可运行于 Unix,linux,windows 中）。

支持基于 IP 或域名的虚拟主机
支持多种方式的 HTTP 认证
集成代理服务器模块
安全 Socket 层(SSL)
能够实时见识服务状态与定制日志
多种模块的支持

**Tomcat**——属于轻量级的 Web 服务软件，一般用于开发和调试 JSP 代码，通常认为 Tomcat 是 Apache 的扩展程序。

总结来说 Nginx 程序作为 Web 服务软件届的后起之秀已经通过自身的努力与优势赢得了大批站长的信赖，例如咱们的《Linux 就该这么学》就是基于 Nginx 服务部署的，不得不说真的很棒！但是 Apache 程序作为老牌的 Web 服务软件因其卓越的稳定性与安全性成为了红帽 RHEL7 系统中默认的网站服务软件，同样也是红帽 RHCSA 与 RHCE 考试认证中避不开的考题。

**9.2 选购服务器主机**

网站是由域名、网页源程序和主机空间组成的，其中主机空间则是用于存放网页源代码并能够将网页内容展示给用户，虽然本小节与 Apache 服务没有直接关系，但如果您想要在互联网中搭建网站并被顺利访问，主机空间一定不能选错。



常见的主机空间包括**虚拟主机**、**VPS**、**云服务器**与**独立服务器**:

**虚拟主机**:在一台服务器中分出一定的磁盘空间供用户放置网站、存放数据等，仅提供基础的网站访问、数据存放与传输流量功能，能够极大的降低用户费用，也几乎不需要管理员维护除网站数据以外的服务，适合小型网站。

**VPS(Virtual Private Server)**:在一台服务器中利用 OpenVZ、Xen 或 KVM 等虚拟化技术模拟出多个"主机"，每个主机都有独立的 IP 地址、操作系统，实现不同 VPS 之间磁盘空间、内存、CPU 资源、进程与系统配置间的完全隔离，管理员可自由使用分配到的主机中的所有资源，所以需要有一定的维护系统的能力，适合小型网站。

**云服务器(ECS)**:是一种整合了计算、存储、网络，能够做到弹性伸缩的计算服务，其使用起来与 VPS 几乎一样，但差别是**云服务器**建立在一组集群服务器中，每个服务器都会保存一个主机的镜像（备份），大大的提升了安全稳定性，另外还具备了灵活性与扩展性，用户只需按使用量付费的即可，适合大中小型网站。

**独立服务器**:这台服务器仅提供给您使用，详细来讲又可以区分为**租用方式**与**托管方式**。

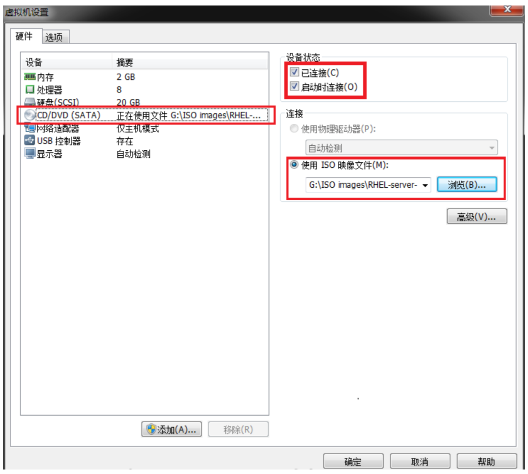
- **租用方式**:用户只需将硬件配置要求告知 IDC 服务商，服务器硬件设备由机房负责维护，运维管理员一般需要自行安装相应的软件并部署网站服务，租期可以为月、季、年，减轻了用户初期对硬件设备的投入，适合大中型网站。
- **托管方式**:用户需要自行购置服务器后交给 IDC 服务供应商的机房进行管理(缴纳管理服务费用)，用户对服务器硬件配置有完全的控制权，自主性强，但需要自行维护、修理服务器硬件设备，适合大中型网站。

另外有必要提醒读者，选择主机空间供应商时请一定要注意看口碑，综合分析再决定购买，某些供应商会有**限制功能**、**强制添加广告**、**隐藏扣费或强制扣费**等恶劣行为，一定一定不要上当!

## 9.3 安装 Apache 服务程序

接下来就要试试动手安装 Apache 服务程序啦，首先按照前面的章节中已经学习过方法挂载光盘设备并 Yum 仓库配置文件。

**第 1 步:在虚拟机软件里选中光盘镜像:**




**第 2 步:将光盘设备挂载到/media/cdrom 目录:**

```
[root@linuxprobe ~]# mkdir -p /media/cdrom
[root@linuxprobe ~]# mount /dev/cdrom /media/cdrom
mount: /dev/sr0 is write-protected, mounting read-only
```

**第 3 步:使用 Vim 编辑器创建 Yum 仓库的配置文件:**

```
[root@linuxprobe ~]# vim /etc/yum.repos.d/rhel7.repo
[rhel7]
name=rhel7
baseurl=file:///media/cdrom
enabled=1
gpgcheck=0
```

**第 4 步:安装 Apache 服务程序:**

需要注意 apache 服务程序的软件包名称叫做 **httpd**，因此直接执行 yum install apache 则是错误的。

[root@linuxprobe ~]# yum install httpd

Loaded plugins: langpacks, product-id, subscription-manager

………………省略部分安装过程信息………………

Installing:

httpd x86_64 2.4.6-17.el7 rhel7 1.2 M

Installing for dependencies:

apr x86_64 1.4.8-3.el7 rhel7 103 k

apr-util x86_64 1.5.2-6.el7 rhel7 92 k

httpd-tools x86_64 2.4.6-17.el7 rhel7 77 k

mailcap noarch 2.1.41-2.el7 rhel7 31 k

Transaction Summary

================================================================================

Install 1 Package (+4 Dependent packages)

Total download size: 1.5 M

Installed size: 4.3 M

Is this ok [y/d/N]: **y**

Downloading packages:

………………省略部分安装过程信息………………

Complete!

因读者们硬件不同或操作错误都可能导致服务安装失败，请耐心再仔细看看操作步骤吧，不要气馁~

您可以将报错信息、屏幕截图、硬件配置与自己的操作过程帖到**“排错答疑区”**，专家们会为您 1 对 1 解答。

**第 5 步:运行 Apache 服务程序并设置为开机启动:**

启动 Apache 服务程序:

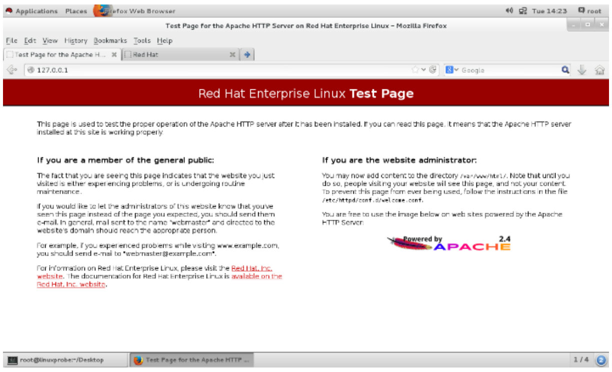
[root@linuxprobe ~]# systemctl start httpd

设置为开机自启动:

[root@linuxprobe ~]# systemctl enable httpd

打开浏览器后键入 http://127.0.0.1，能看到默认页面了吗:

[root@linuxprobe ~]# firefox



**9.4 配置服务文件参数**

慢着!先别激动!!刚刚学会的安装和运行只是学习 Apache 服务成功路上的第一步,现在来了解下各个 httpd 服务目录都是干嘛用的吧:

| | |
|---|---|
| 服务目录 | /etc/httpd |
| 配置文件 | /etc/httpd/conf/httpd.conf |
| 网站数据目录 | /var/www/html |
| 访问日志 | /var/log/httpd/access_log |
| 错误日志 | /var/log/httpd/error_log |

打开 Apache 服务程序的配置文件:

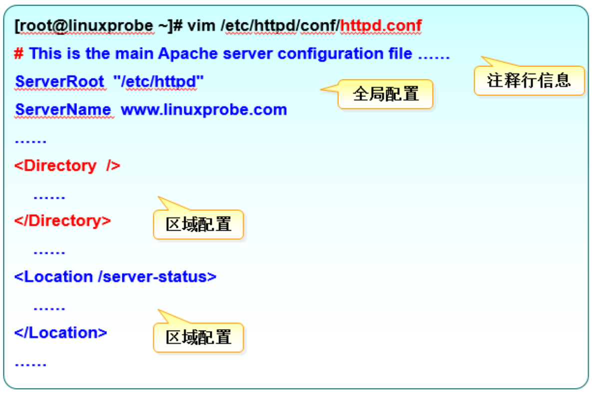
[root@linuxprobe ~]# vim /etc/httpd/conf/httpd.conf

初次看到配置文件可真的吓了一跳，353 行!这没有一周研究不完吧!

其实吓唬你们的了,所有以#号开始的叫注释行，这些只是描述介绍而已，真正的参数有:

| | |
|---|---|
| ServerRoot | 服务目录 |
| ServerAdmin | 管理员邮箱 |
| User | 运行服务的用户 |
| Group | 运行服务的用户组 |
| ServerName | 网站服务器的域名 |
| DocumentRoot | 网站数据目录 |
| Listen | 监听的 IP 地址与端口号 |
| DirectoryIndex | 默认的索引页页面 |
| ErrorLog | 错误日志文件 |
| CustomLog | 访问日志文件 |
| Timeout | 网页超时时间,默认为 300 秒. |
| Include | 需要加载的其他文件 |

Apache 服务程序的配置文件内容分为三种类型：“**注释行信息**”，“**全局配置**”，“**区域配置**”。



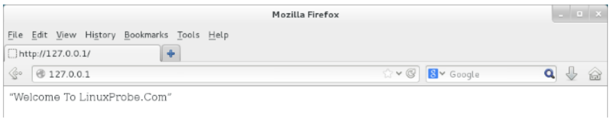

默认的网站数据是存放在/var/www/html 目录中的，首页名称是 **index.html**，来动手写入一个文件替换到默认页面吧。

使用 echo 命令将指定的字符写入到网站数据目录中的 index.html 文件中：

[root@linuxprobe ~]# echo "Welcome To LinuxProbe.Com" > /var/www/html/index.html

再次打开浏览器，键入 http://127.0.0.1，好棒，成功了！

[root@linuxprobe ~]# firefox




这样一试果然成功了，原来真不是很难，信心大涨！要想将网站数据放在/home/wwwroot 目录，该如何操作呢？

**编辑 Apache 服务程序的主配置文件：**

[root@linuxprobe ~]# vim /etc/httpd/conf/httpd.conf

将在 119 行的 **DocumentRoot** 参数修改为”**/home/wwwroot**“,再把在 123 行的”**/var/www**“修改为”**/home/wwwroot**“

建立网站数据目录：

[root@linuxprobe ~]# mkdir /home/wwwroot

创建首页文件：

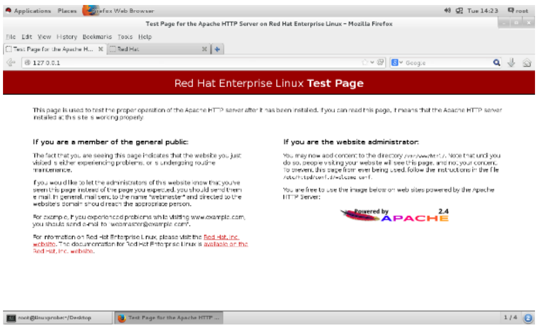
[root@linuxprobe ~]# echo "The New Web Directory" > /home/wwwroot/index.html

重新启动 Apache 服务：

[root@linuxprobe ~]# systemctl restart httpd

再来打开浏览器看下效果吧，依然是键入 http://127.0.0.1：

[root@linuxprobe ~]# firefox




好奇怪!!为什么会是默认页面？只有首页页面不存在或有问题才会显示 Apache 服务程序的默认页面啊。

那么进一步来访问”http://127.0.0.1/index.html“，怎么样？惊讶到了吗？访问页面的行为是被禁止的。

Forbidden

You don’t have permission to access /index.html on this server.

我们的操作与刚刚的前面的实验一样啊，但这次的访问行为会被禁止呢？这就要先了解下 SElinux 啦。

**9.5 强制访问控制安全子系统**

SELinux 全称为 Security-Enhanced Linux 是美国国家安全局在 Linux 社区帮助下开发的一个强制访问控制的安全子系统，SELinux 属于 MAC 强制访问控制（Mandatory Access Control）——即让系统中的各个服务进程都受到约束，即仅能访问到所需要的文件。

以本人的亲身经历不得不说国内很多运维人员对 SELinux 的理解不深，导致该功能在很多服务器中直接被禁用。

模式一: enforcing – 安全策略强制启用模式，将会拦截服务的不合法请求。

模式二: permissive – 遇到服务越权访问只会发出警告而不强制拦截。

模式三: disabled – 对于越权的行为不警告，也不拦截。

有时关闭 SELinux 后确实能够减少报错几率，但这极其的不推荐并且**本书实验环境均为开启状态**，确保您的 SELinux 服务是默认启用的:

第 1 步:切换到 SELinux 服务的配置文件目录”**/etc/selinux**“。

第 2 步:编辑 **config** 文件将模式改为强制启用，记得保存哦！

如果发现 SELINUX=permissive 或 disabled 那就赶紧改过来吧：

[root@linuxprobe ~]# cd /etc/selinux

[root@linuxprobe selinux]# vim config

SELINUX=enforcing

此时可以来查询下当前的 SELinux 服务状态：

[root@linuxprobe ~]# getenforce

Enforcing

**9.6 允许 SELinux 策略**

哦~~原来如此，为了确认是这个讨厌的 SELinux 服务在捣鬼，我们来先关闭它试试吧：

[root@linuxprobe ~]# setenforce 0

检查状态，现在已经是“禁止模式”了：

[root@linuxprobe ~]# getenforce

Permissive

打开浏览器再键入 http://127.0.0.1，果然成功了！

[root@linuxprobe wwwroot]#firefox




果然是因为没有配置好 SELinux 服务，再次开启它吧：

[root@linuxprobe ~]# setenforce 1

刚刚浏览器里报错说“**禁止，你没有访问 index.html 文件的权限**”，那怎么开启 SELinux 的允许策略呢？

SELinux 安全策略包括域和安全上下文：

**SELinux 域**:对进程资源进行限制（查看方式:ps -Z）

**SELinux 安全上下文**:对系统资源进行限制（查看方式:ls -Z）

使用”ls -Z“命令检查下新旧网站数据目录的 SELinux 安全上下文有何不同吧:

[root@linuxprobe ~]# ls -Zd /var/www/html

drwxr-xr-x. root root **system_u:object_r:httpd_sys_content_t:s0** /var/www/html

[root@linuxprobe ~]# ls -Zd /home/wwwroot

drwxrwxrwx. root root **unconfined_u:object_r:home_root_t:s0** /home/wwwroot

SELinux 安全上下文是由冒号间隔的四个字段组成的，以原始网站数据目录的安全上下文为例分析下吧:

**用户段**:root 表示 root 账户身份，user_u 表示普通用户身份，system_u 表示系统进程身份。

**角色段**:object_r 是文件目录角色，system_r 是一般进程角色。

**类型段**:进程和文件都有一个类型用于限制存取权限。

解决办法就是将当前网站目录”/home/wwwroot“的安全上下文修改成 system_u:object_r:httpd_sys_content_t:s0 就可以啦~

**semanage 命令用于查询与修改 SELinux 的安全上下文，格式为：“semanage [选项] [文件]”。**

| 参数 | 作用 |
| --- | --- |
| -l | 查询 |
| -a | 增加 |
| -m | 修改 |
| -d | 删除 |

**restorecon 命令用于恢复 SELinux 文件安全上下文，格式为：“restorecon [选项] [文件]”。**

| -i | 忽略不存在的文件 |
| --- | --- |
| -e | 排除目录 |
| -R | 递归处理(针对目录使用) |
| -v | 显示详细的过程 |
| -F | 强制恢复 |

修改网站数据目录的安全上下文:

[root@linuxprobe ~]# semanage fcontext -a -t httpd_sys_content_t /home/wwwroot

修改网站数据的安全上下文（*代表所有文件或目录）:

[root@linuxprobe ~]# semanage fcontext -a -t httpd_sys_content_t /home/wwwroot/*

这样操作后查看到 SELinux 安全上下文依然没有改变，不要着急，再执行下 restorecon 命令即可:

[root@linuxprobe ~]# restorecon -Rv /home/wwwroot/

再来刷新浏览器后看到正常页面:

[root@linuxprobe ~]# firefox




真可谓是一波三折，原本以为将 Apache 服务配置妥当就大功告成，结果却受到了 SELinux 安全上下文的限制，看来真是要细心才行。

## 9.7 个人用户主页功能

Apache 服务程序中有个默认未开启的个人用户主页功能，能够为所有系统内的用户生成个人网站，确实很实用哦~

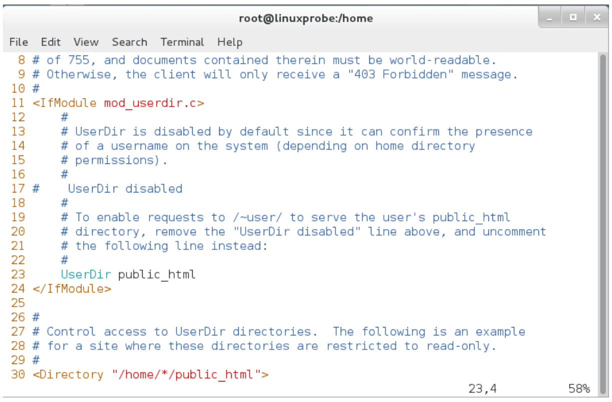
**第 1 步:开启个人用户主页功能：**

[root@linuxprobe ~]# vim /etc/httpd/conf.d/userdir.conf

将第 17 行的 UserDir disabled 前加一个#，代表该行被注释掉，不再起作用。

将第 23 行的 UserDir public_html 前的#号去除，表示该行被启用。

注意:UserDir 参数表示的是需要在用户家目录中创建的网站数据目录的名称(即 public_html)




重启 Apache 服务程序：

[root@linuxprobe ~]# systemctl restart httpd

**第 2 步:创建个人用户网站数据。**

切换至普通会员 linuxprobe 的家目录：

[root@linuxprobe home]# su - linuxprobe

Last login: Fri May 22 13:17:37 CST 2015 on :0

创建网站数据目录 public_html:

[linuxprobe@linuxprobe ~]$ mkdir public_html

写入首页文件内容：

[linuxprobe@linuxprobe ~]$ echo "This is linuxprobe's website" > public_html/index.html

给予网站目录 755 的访问权限：

[linuxprobe@linuxprobe ~]$ chmod -Rf 755 ./

我们打开浏览器，访问地址为"**http://127.0.0.1/~用户名**"，不出意外果然是报错页面，肯定是 SELinux 服务在捣蛋。




**第 3 步:设置 SELinux 允许策略。**

这次报错并不是因为用户家的网站数据目录 SELinux 安全上下文没有设置了，而是因为 SELinux 默认就不允许 Apache 服务个人用户主页这项功能。

**getsebool 命令用于查询所有 SELinux 规则的布尔值，格式为："getsebool -a"。**

SELinux 策略布尔值:只有 0/1 两种情况，0 或 off 为禁止，1 或 on 为允许。

**setsebool 命令用于修改 SElinux 策略内各项规则的布尔值，格式为："setsebool [选项] 布尔值=[0|1]"。**

| 参数 | 作用 |
| --- | --- |
| -P | 永久生效 |

查看并搜索所有与家目录有关的 SELinux 策略：

[root@linuxprobe ~]# getsebool -a | grep home
  ftp_home_dir --> off
  git_cgi_enable_homedirs --> off
  git_system_enable_homedirs --> off
  httpd_enable_homedirs --> off
  mock_enable_homedirs --> off
  mpd_enable_homedirs --> off
  openvpn_enable_homedirs --> on
  samba_create_home_dirs --> off
  samba_enable_home_dirs --> off
  sftpd_enable_homedirs --> off
  sftpd_write_ssh_home --> off
  spamd_enable_home_dirs --> on
  ssh_chroot_rw_homedirs --> off
  tftp_home_dir --> off
  use_ecryptfs_home_dirs --> off
  use_fusefs_home_dirs --> off
  use_nfs_home_dirs --> off
  use_samba_home_dirs --> off
  xdm_write_home --> off

将个人用户网站功能策略设置为允许：

[root@linuxprobe ~]# setsebool -P httpd_enable_homedirs=on

刷新浏览器访问 linuxprobe 用户的个人网站，果然成功了：



**第 4 步:增加密码安全验证。**

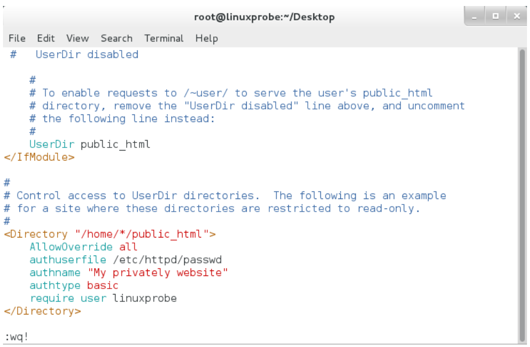
有时候并不希望所有人都可以留意访问到自己的个人网站，那就可以使用 Apache 密码口令验证功能增加一道安全防护吧。

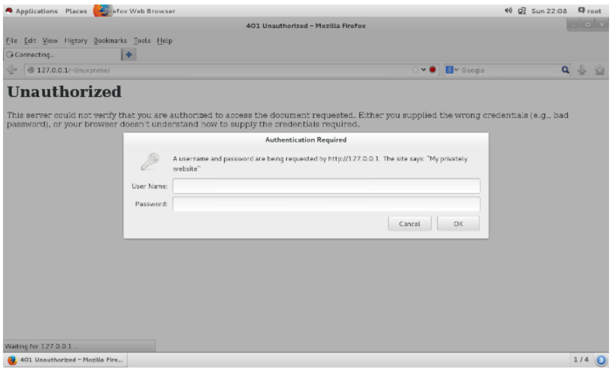
使用 htpasswd 命令生成密码数据库（-c 参数用于第一次生成）：

[root@linuxprobe ~]# htpasswd -c /etc/httpd/passwd linuxprobe
New password:
Re-type new password:
Adding password for user linuxprobe

编辑配置文件开启密码验证（具体参数见下图）：

[root@linuxprobe ~]# vim /etc/httpd/conf.d/userdir.conf






如果口令输入错误会直接禁止访问:

Unauthorized

This server could not verify that you are authorized to access the document requested. Either you supplied the wrong credentials (e.g., bad password), or your browser doesn’t understand how to supply the credentials required.

这里的 User Name 是 linuxprobe，密码并非该用户的系统密码，而是 htpasswd 命令创建的网站密码，不要搞混哦~

## 9.8 虚拟网站主机功能

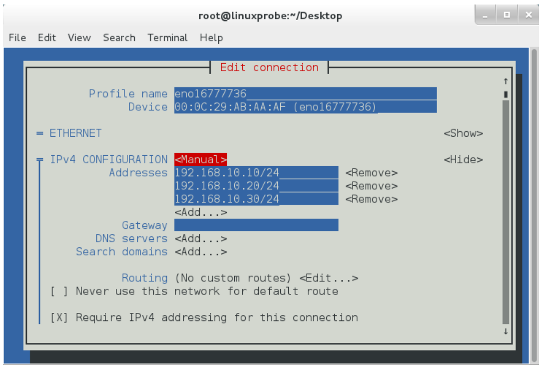
Apache 的虚拟主机功能 (Virtual Host) 是可以让一台服务器基于 IP、主机名或端口号实现提供多个网站服务的技术。虚拟主机功能的操作步骤都很简单，但可能比较难理解其中的原理，一旦搭建出实验环境，你就一定会明白了。[附件]



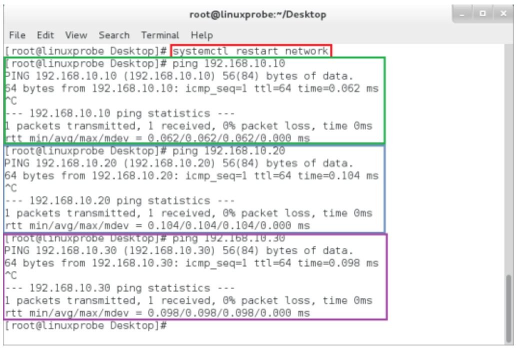

### 9.8.1 基于 IP 地址

这种情况很常见:一台服务器拥有多个 IP 地址，当用户访问不同 IP 地址时显示不同的网站页面。

**第 1 步:使用 nmtui 命令为网卡添加多个 IP 地址（192.168.10.10/20/30）:**




重新启动网卡设备后使用 ping 命令检查是否配置正确（**这项很重要，一定要测试好再进行下一步!**）。



**第 2 步:分别创建网站数据目录。**

在/home/wwwroot 目录下分别创建三个网站数据目录:

[root@linuxprobe ~]# mkdir -p /home/wwwroot/10
[root@linuxprobe ~]# mkdir -p /home/wwwroot/20
[root@linuxprobe ~]# mkdir -p /home/wwwroot/30

分别在这三个网站数据目录中写入主页文件，内容为该网站的 IP 地址:

[root@linuxprobe ~]# echo "IP:192.168.10.10" > /home/wwwroot/10/index.html
[root@linuxprobe ~]# echo "IP:192.168.10.20" > /home/wwwroot/20/index.html
[root@linuxprobe ~]# echo "IP:192.168.10.30" > /home/wwwroot/30/index.html

**第 3 步:在配置文件中描述基于 IP 地址的虚拟主机。**

**<VirtualHost 192.168.10.10>**
DocumentRoot /home/wwwroot/10
ServerName www.linuxprobe.com
<Directory /home/wwwroot/10 >
AllowOverride None
Require all granted
</Directory>
**</VirtualHost>**
**<VirtualHost 192.168.10.20>**
DocumentRoot /home/wwwroot/20
ServerName bbs.linuxprobe.com
<Directory /home/wwwroot/20 >
AllowOverride None
Require all granted
</Directory>
**</VirtualHost>**
**<VirtualHost 192.168.10.30>**
DocumentRoot /home/wwwroot/30
ServerName tech.linuxprobe.com
<Directory /home/wwwroot/30 >
AllowOverride None
Require all granted
</Directory>
**</VirtualHost>**

**第 4 步:修改网站数据目录的 SELinux 安全上下文。**

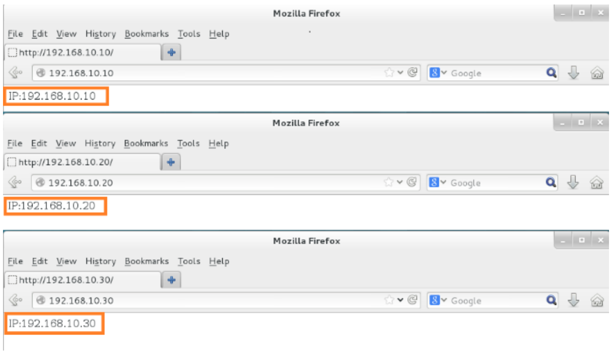
需要分别修改网站数据目录以及网页文件的 SELinux 安全上下文:

[root@linuxprobe ~]# semanage fcontext -a -t httpd_sys_content_t /home/wwwroot
[root@linuxprobe ~]# semanage fcontext -a -t httpd_sys_content_t /home/wwwroot/10
[root@linuxprobe ~]# semanage fcontext -a -t httpd_sys_content_t /home/wwwroot/10/*
[root@linuxprobe ~]# semanage fcontext -a -t httpd_sys_content_t /home/wwwroot/20
[root@linuxprobe ~]# semanage fcontext -a -t httpd_sys_content_t /home/wwwroot/20/*
[root@linuxprobe ~]# semanage fcontext -a -t httpd_sys_content_t /home/wwwroot/30
[root@linuxprobe ~]# semanage fcontext -a -t httpd_sys_content_t /home/wwwroot/30/*

立即恢复 SELinux 安全上下文：

[root@linuxprobe ~]# restorecon -Rv /home/wwwroot

**第 5 步:分别访问 192.168.10.10/20/30 验证结果：**

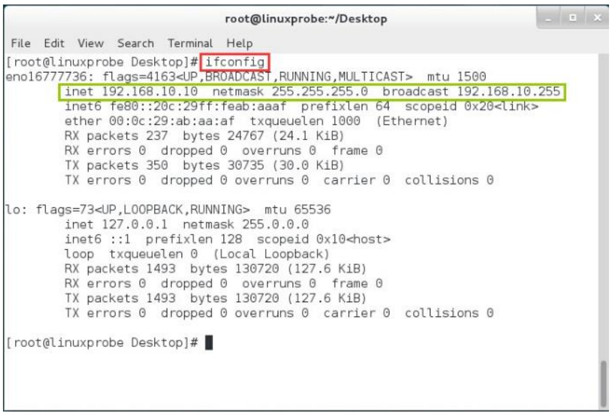



**请注意:当您完成本实验后请还原虚拟机快照再进行下一个实验，否则可能导致配置文件冲突而报错。**

### 9.8.2 基于主机名

当服务器无法为每个网站都分配到独立 IP 地址时，可以试试让 Apache 服务程序自动识别来源主机名或域名然后跳转到指定的网站。

**第 1 步:配置网卡 IP 地址与 hosts 文件。**




hosts 文件作用是定义 IP 地址与主机名的映射关系，**即强制将某个主机名地址解析到指定的 IP 地址**。

[root@linuxprobe ~]# vim /etc/hosts

//每行只能写一条，格式为 IP 地址+空格+主机名（域名）。

192.168.10.10 www.linuxprobe.com

192.168.10.10 bbs.linuxprobe.com

192.168.10.10 tech.linuxprobe.com

**第 2 步:分别创建网站数据目录：**

[root@linuxprobe ~]# mkdir -p /home/wwwroot/www

[root@linuxprobe ~]# mkdir -p /home/wwwroot/bbs

[root@linuxprobe ~]# mkdir -p /home/wwwroot/tech

**分别在网站目录中写入不同的首页文件：**

[root@linuxprobe ~]# echo "WWW.linuxprobe.com" > /home/wwwroot/www/index.html

[root@linuxprobe ~]# echo "BBS.linuxprobe.com" > /home/wwwroot/bbs/index.html

[root@linuxprobe ~]# echo "TECH.linuxprobe.com" > /home/wwwroot/tech/index.html

**第 3 步:在配置文件中描述基于主机名称的虚拟主机。**

编辑主配置文件(/etc/httpd/conf/httpd.conf)，在主配置文件的末尾按下面格式定义虚拟主机信息：

```
<VirtualHost 192.168.10.10>
DocumentRoot "/home/wwwroot/www"
ServerName "www.linuxprobe.com"
<Directory "/home/wwwroot/www">
AllowOverride None
Require all granted
</directory>
</VirtualHost>

<VirtualHost 192.168.10.10>
DocumentRoot "/home/wwwroot/bbs"
ServerName "bbs.linuxprobe.com"
<Directory "/home/wwwroot/bbs">
AllowOverride None
Require all granted
</Directory>
</VirtualHost>

<VirtualHost 192.168.10.10>
DocumentRoot "/home/wwwroot/tech"
ServerName "tech.linuxprobe.com"
<Directory "/home/wwwroot/tech">
AllowOverride None
Require all granted
</directory>
</VirtualHost>
```

**读者们可以直接复制上面的参数到主配置文件(/etc/httpd/conf/httpd.conf)的末尾然后重启 apache 网站服务程序。**

因为在红帽 RHCSA、RHCE 或 RHCA 考试后都要重启您的实验机再执行判分脚本。

所以请读者在日常工作中也要记得将需要的服务加入到开机启动项中:"systemctl enable httpd"。

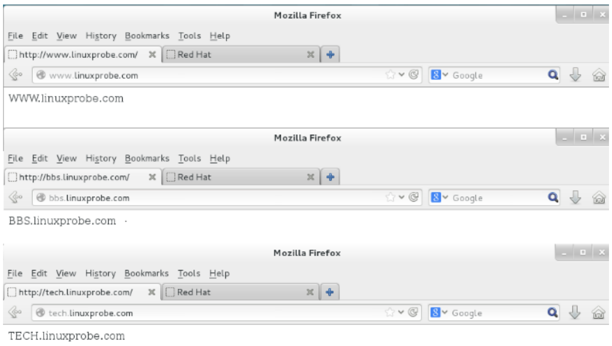
**第 4 步:修改网站数据目录的 SELinux 安全上下文:**

```
[root@linuxprobe ~]# semanage fcontext -a -t httpd_sys_content_t /home/wwwroot
[root@linuxprobe ~]# semanage fcontext -a -t httpd_sys_content_t /home/wwwroot/www
[root@linuxprobe ~]# semanage fcontext -a -t httpd_sys_content_t /home/wwwroot/www/*
[root@linuxprobe ~]# semanage fcontext -a -t httpd_sys_content_t /home/wwwroot/bbs
[root@linuxprobe ~]# semanage fcontext -a -t httpd_sys_content_t /home/wwwroot/bbs/*
[root@linuxprobe ~]# semanage fcontext -a -t httpd_sys_content_t /home/wwwroot/tech
[root@linuxprobe ~]# semanage fcontext -a -t httpd_sys_content_t /home/wwwroot/tech/*
```

让新的 SELinux 安全上下文立即生效：

```
[root@linuxprobe ~]# restorecon -Rv /home/wwwroot/
```

**第 5 步:分别访问网站验证结果**

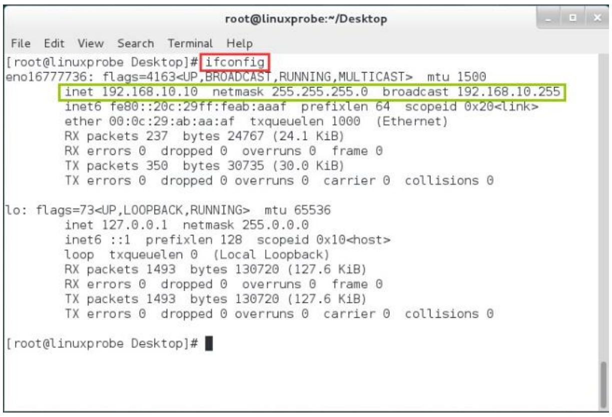



**请注意:当您完成本实验后请还原虚拟机快照再进行下一个实验，否则可能导致配置文件冲突而报错。**

### 9.8.2 基于端口号

我们可以让服务器开启多个服务端口后，然后让用户能够通过访问服务器的指定端口来找到想要的网站。

**第 1 步:配置服务器的 IP 地址:**




**第 2 步:分别创建网站数据目录。**

分别创建端口为 6111,6222 的网站数据目录:

[root@linuxprobe ~]# mkdir -p /home/wwwroot/6111

[root@linuxprobe ~]# mkdir -p /home/wwwroot/6222

分别在网站数据目录中写入不同内容的主页文件:

[root@linuxprobe ~]# echo "port:6111" > /home/wwwroot/6111/index.html

[root@linuxprobe ~]# echo "port:6222" > /home/wwwroot/6222/index.html

**第 3 步:在配置文件中描述基于端口号的虚拟主机。**

编辑主配置文件(**/etc/httpd/conf/httpd.conf**)，找到约在 42 行的 Listen 80，并在下面追加:

Listen 6111

Listen 6222

然后在主配置文件的末尾按下面格式定义虚拟主机信息:

**<VirtualHost 192.168.10.10:6111>**

DocumentRoot “/home/wwwroot/6111”

ServerName www.linuxprobe.com

<Directory “/home/wwwroot/6111” >

AllowOverride None

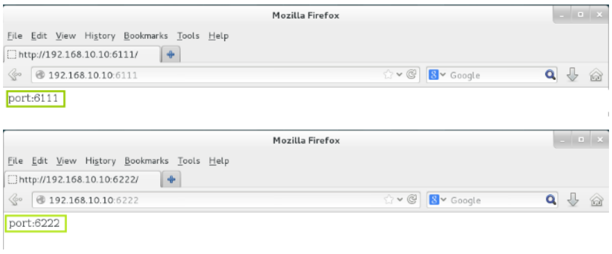
Require all granted

</Directory>

**</VirtualHost>**
**<VirtualHost 192.168.10.10:6222>**
DocumentRoot “/home/wwwroot/6222”
ServerName bbs.linuxprobe.com
<Directory “/home/wwwroot/6222”>
AllowOverride None
Require all granted
</Directory>
**</VirtualHost>**

**读者们可以直接复制上面的参数到主配置文件(/etc/httpd/conf/httpd.conf)的末尾然后重启 apache 网站服务程序。**
因为在红帽 RHCSA、RHCE 或 RHCA 考试后都要重启您的实验机再执行判分脚本。
所以请读者在日常工作中也要记得将需要的服务加入到开机启动项中:”systemctl enable httpd“。
**什么!竟然报错了:**
Job for httpd.service failed. See 'systemctl status httpd.service' and 'journalctl -xn' for details.
这是因为 SELinux 服务检测到 6111 与 6222 端口原本并不属于 Apache 服务端口，但现在却被以 Apache 的名义监听了.
**第 4 步:修改网站数据目录的 SELinux 安全上下文并允许端口监听。**
修改网站数据目录的安全上下文:
[root@linuxprobe ~]# semanage fcontext -a -t httpd_user_content_t /home/wwwroot
[root@linuxprobe ~]# semanage fcontext -a -t httpd_user_content_t /home/wwwroot/6111
[root@linuxprobe ~]# semanage fcontext -a -t httpd_user_content_t /home/wwwroot/6111/*
[root@linuxprobe ~]# semanage fcontext -a -t httpd_user_content_t /home/wwwroot/6222
[root@linuxprobe ~]# semanage fcontext -a -t httpd_user_content_t /home/wwwroot/6222/*
让新的 SElinux 安全上下文立即生效:
[root@linuxprobe ~]# restorecon -Rv /home/wwwroot/
使用 semanage 命令搜索在 SELinux 系统中有关 http 服务的端口号:
[root@linuxprobe ~]# semanage port -l| grep http
http_cache_port_t tcp 8080, 8118, 8123, 10001-10010
http_cache_port_t udp 3130
http_port_t tcp **80, 81, 443, 488, 8008, 8009, 8443, 9000**
pegasus_http_port_t tcp 5988
pegasus_https_port_t tcp 5989
默认包括 80,81,443,488,8008,8009,8443,9000 却没有咱们定义的端口号，那么添加进去就可以了:
[root@linuxprobe ~]# semanage port -a -t http_port_t -p tcp 6111
[root@linuxprobe ~]# semanage port -a -t http_port_t -p tcp 6222
再来看下 SELinux 的端口规则（已经添加成功了）:
[root@linuxprobe ~]# semanage port -l| grep http
http_cache_port_t tcp 8080, 8118, 8123, 10001-10010
http_cache_port_t udp 3130
http_port_t tcp **6222, 6111, 80, 81, 443, 488, 8008, 8009, 8443, 9000**
pegasus_http_port_t tcp 5988
pegasus_https_port_t tcp 5989
再次尝试启动 Apache 网站服务程序就没有问题了:
[root@linuxprobe ~]# systemctl restart httpd

**第 5 步:分别访问网站验证结果:**




**请注意:当您完成本实验后请还原虚拟机快照再进行下一个实验，否则可能导致配置文件冲突而报错。**

## 9.9 Apache 的访问控制

我们还可以**基于主机名、IP 地址**以及**客户端特征**做 Apache 网页资源的访问控制，常用的指令有:

**Order(排序)，Allow(允许)，Deny(拒绝)，Satisfy(满足)。**

其中 Order 指令用于定义 Allow 或 Deny 起作用的顺序，分别实现了允许或者拒绝某个主机访问服务器网页资源。

**匹配原则为:**按顺序匹配规则并执行，若未匹配成功则执行后面的执行。

比如说”Order Allow,Deny“则代表着先将客户端与允许规则进行对比，若匹配成功则允许访问，反之则直接拒绝。

创建网站数据目录和首页文件:

[root@localhost ~]# mkdir /var/www/html/server

[root@localhost ~]# echo "Successful" > /var/www/html/server/index.html

**根据浏览器的变量特征，只允许 IE 浏览器访问本网站数据。**

[root@localhost ~]# vim /etc/httpd/conf/httpd.conf

//在大约 129 行的地方添加参数.

<Directory "/var/www/html/server">

SetEnvIf User-Agent “Internet Explorer” ie=1

Order allow,deny

Allow from env=ie

保存配置文件后记得重启服务(systemctl restart httpd)，然后用 Firefox 浏览器尝试访问网站页面:




那么如果希望仅允许火狐浏览器访问本页面，请将配置文件修改为:

<Directory “/var/www/html/server”>

SetEnvIf User-Agent “Firefox” ff=1

Order allow,deny

Allow from env=ff

</Directory>

**根据来访源地址，仅限 192.168.10.10 的主机访问本网站。**

此时我们就需要两台主机来完成实验了，请配置主机 IP 地址后能够互相通信。

| 主机名称 | 操作系统 | IP 地址 |
| --- | --- | --- |
| 本地主机 | 红帽 RHEL7 操作系统 | 192.168.10.10 |
| 远程主机 | 红帽 RHEL7 操作系统 | 192.168.10.20 |

[root@localhost ~]# vim /etc/httpd/conf/httpd.conf

//在大约 129 行的地方添加参数.

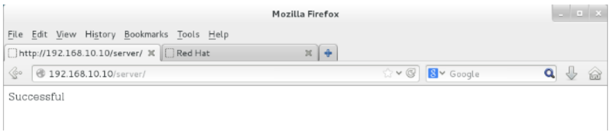
Order allow,deny

Allow from 192.168.10.20

保存配置文件后记得重启服务(systemctl restart httpd)，然后用 Firefox 浏览器尝试访问网站页面：




然后再使用远程主机（192.168.10.20）尝试访问页面，顺利的成功了：



# 第 10 章 使用 Vsftpd 服务传输文件。

**章节简述：**

本章节先通过介绍文件传输协议来帮助读者理解 FTP 协议的用处，安装 vsftpd 服务程序并逐条分析服务文件的配置参数。

完整演示 vsftpd 服务匿名访问模式、本地用户模式及虚拟用户模式的配置方法，介绍 PAM 可插拔式认证模块的原理与认证流程。

通过配置 vsftpd 服务程序，进一步的锻炼了读者 SELinux 服务策略、安全上下文以及防火墙的配置与排错能力。

## 10.1 文件传输协议

文件传输协议（FTP,File Transfer Protocol），即能够让用户在互联网中上传、下载文件的文件协议，而 FTP 服务器就是支持 FTP 传输协议的主机，要想完成文件传输则需要 FTP 服务端和 FTP 客户端的配合才行。

通常用户使用 FTP 客户端软件向 FTP 服务器发起连接并发送 FTP 指令，服务器收到用户指令后将执行结果返回客户端。




**FTP 协议占用两个端口号：**

21 端口:命令控制，用于接收客户端执行的 FTP 命令。

20 端口:数据传输，用于上传、下载文件数据。

**FTP 数据传输的类型：**

主动模式:FTP 服务端主动向 FTP 客户端发起连接请求。

被动模式:FTP 服务端等待 FTP 客户端的连接请求。

## 10.2 安装 vsftpd 服务程序

Vsftpd 即"Very Secure FTP Daemon"是一款运行在类 Unix 操作系统的 FTP 服务端程序，Vsftpd 主打的是安全性、完全开源及免费、速率高、支持 IPv6、虚拟用户功能等等其他 FTP 服务端软件不具备的功能。

安装 vsftpd 服务程序包：

```
[root@linuxprobe ~]# yum install vsftpd -y
Loaded plugins: langpacks, product-id, subscription-manager
…………………省略部分安装过程………………
---> Package vsftpd.x86_64 0:3.0.2-9.el7 will be installed
--> Finished Dependency Resolution
…………………省略部分安装过程…………………
Installed:
vsftpd.x86_64 0:3.0.2-9.el7
Complete!
```

清空默认的防火墙默认规则：

```
[root@linuxprobe ~]# iptables -F
```

保存清空后的防火墙规则表：

```
[root@linuxprobe ~]# service iptables save
```

**Vsftpd 的程序与配置文件：**

| 主程序 | /usr/sbin/vsftpd |
| --- | --- |
| 用户禁止登陆列表 | /etc/vsftpd/ftpusers<br>/etc/vsftpd/user_list |
| 主配置文件 | /etc/vsftpd/vsftpd.conf |

先来分析下 vsftpd 程序的主配置文件吧：

[root@linuxprobe ~]# cat /etc/vsftpd/vsftpd.conf

主配置文件长达 123 行，但大部分是以#号开始的，这些都是注释信息，我们可以过滤掉它们。

备份 vsftpd 的主配置文件：

[root@linuxprobe ~]# mv /etc/vsftpd/vsftpd.conf /etc/vsftpd/vsftpd.conf_bak

过滤掉所有包含#号的行，并将过滤结果写回到 vsftpd.conf 文件中：

[root@linuxprobe ~]# grep -v "#" /etc/vsftpd/vsftpd.conf_bak > /etc/vsftpd/vsftpd.conf

此时再分析下 vsftpd 程序的主配置文件吧：

[root@linuxprobe ~]# **cat /etc/vsftpd/vsftpd.conf**

anonymous_enable=YES
local_enable=YES
write_enable=YES
local_umask=022
dirmessage_enable=YES
xferlog_enable=YES
connect_from_port_20=YES
xferlog_std_format=YES
listen=NO
listen_ipv6=YES
pam_service_name=vsftpd
userlist_enable=YES
tcp_wrappers=YES

**vsftpd 程序配置文件参数的作用：**

| 参数 | 作用 |
| --- | --- |
| listen=[YES\|NO] | 是否以独立运行的方式监听服务。 |
| listen_address=IP 地址 | 设置要监听的 IP 地址。 |
| listen_port=21 | 设置 FTP 服务的监听端口。 |
| download_enable＝[YES\|NO] | 是否允许下载文件。 |
| userlist_enable=[YES\|NO]<br>userlist_deny=[YES\|NO] | 是否启用“禁止登陆用户名单”。 |
| max_clients=0 | 最大客户端连接数，0 为不限制。 |

| | |
|---|---|
| max_per_ip=0 | 同一 IP 地址最大连接数，0 位不限制。 |
| anonymous_enable=[YES\|NO] | 是否允许匿名用户访问。 |
| anon_upload_enable=[YES\|NO] | 是否允许匿名用户上传文件。 |
| anon_umask=022 | 匿名用户上传文件的 umask 值。 |
| anon_root=/var/ftp | 匿名用户的 FTP 根目录。 |
| anon_mkdir_write_enable=[YES\|NO] | 是否允许匿名用户创建目录。 |
| anon_other_write_enable=[YES\|NO] | 是否开放匿名用户其他写入权限。 |
| anon_max_rate=0 | 匿名用户最大传输速率(字节)，0 为不限制。 |
| local_enable=[YES\|NO] | 是否允许本地用户登陆 FTP。 |
| local_umask=022 | 本地用户上传文件的 umask 值。 |
| local_root=/var/ftp | 本地用户的 FTP 根目录。 |
| chroot_local_user=[YES\|NO] | 是否将用户权限禁锢在 FTP 目录，更加的安全。 |
| local_max_rate=0 | 本地用户最大传输速率(字节)，0 为不限制。 |

### 10.3 Vsftpd 的验证方式

vsftpd 程序提供的 FTP 服务可选认证方式，分别为**匿名访问**、**本地用户**和**虚拟用户**:

匿名访问:任何人无需验证口令即可登入 FTP 服务端。

本地用户:使用 FTP 服务器中的用户、密码信息。

虚拟用户:创建独立的 FTP 帐号资料。

顾名思义匿名访问就是所有人均可随意登入 FTP 服务，这样自然会产生安全问题，一般用于存放公开的数据。

而本地用户与虚拟用户则需要用户提供帐号及口令后才能登入 FTP 服务，更加的安全，而虚拟用户则是最安全的。

下面的实验环节将使用两台红帽 RHEL7 系统的主机，读者需要提前配置网卡的 IP 地址等信息:

| 主机名称 | 操作系统 | IP 地址 |
|---|---|---|
| FTP 服务端 | 红帽 RHEL7 操作系统 | 192.168.10.10 |
| FTP 客户端 | 红帽 RHEL7 操作系统 | 192.168.10.20 |

### 10.3.1 匿名访问模式

FTP 匿名访问模式是比较不安全的服务模式，尤其在真实的工作环境中千万不要存放敏感的数据，以免泄露。

vsftpd 程序默认已经允许匿名访问模式，我们要做的就是开启匿名用户的上传和写入权限，写入下面的参数：

[root@linuxprobe ~]# vim /etc/vsftpd/vsftpd.conf

| 参数 | 作用 |
| --- | --- |
| anonymous_enable=YES | 允许匿名访问模式。 |
| anon_umask=022 | 匿名用户上传文件的 umask 值。 |
| anon_upload_enable=YES | 允许匿名用户上传文件 |
| anon_mkdir_write_enable=YES | 允许匿名用户创建目录 |
| anon_other_write_enable=YES | 允许匿名用户修改目录名或删除目录 |

确认填写正确后保存并退出 vsftpd.conf 文件，然后重启 vsftpd 服务程序并设置为开机自启动。

[root@linuxprobe ~]# *systemctl restart vsftpd*

[root@linuxprobe ~]# *systemctl enable vsftpd*

ln -s '/usr/lib/systemd/system/vsftpd.service' '/etc/systemd/system/multi-user.target.wants/vsftpd.service

因为在红帽 RHCSA、RHCE 或 RHCA 考试后都要重启您的实验机再执行判分脚本。

所以请读者在日常工作中也要记得将需要的服务加入到开机启动项中:”systemctl enable vsftpd“。

如果重启 vsftpd 服务程序时没有报错，此时便可以使用 FTP 客户机(192.168.10.20)尝试登入 FTP 服务了。

**ftp 命令用于使用 FTP 服务，格式为：“ftp [参数] [FTP 主机]”。**

红帽 RHEL7 系统中 ftp 命令默认没有安装，请执行”yum install ftp -y“即可安装完毕。

**在客户端尝试登入 FTP 服务：**

[root@linuxprobe ~]# **ftp 192.168.10.10**

Connected to 192.168.10.10 (192.168.10.10).

220 (vsFTPd 3.0.2)

Name (192.168.10.10:root): **anonymous**

331 Please specify the password.

Password:**敲击回车**

230 Login successful.

Remote system type is UNIX.

Using binary mode to transfer files.

ftp> **cd pub**

250 Directory successfully changed.

ftp> **mkdir files**

550 Permission denied.

上面操作中已经将防火墙规则清空，在 vsftpd.conf 文件中也已经允许匿名用户创建目录与写入权限，那怎么会被拒绝了呢？

**这里建议读者先不要往下看，**思考后用自己的方法解决下这个问题，长期这样你的 Linux 的排错能力一定会练出来的。

回想前面的参数细节，匿名访问模式的 FTP 根目录为/var/ftp：

[root@linuxprobe ~]# ls -ld /var/ftp/pub

drwxr-xr-x. 3 root root 16 Jul 13 14:38 /var/ftp/pub

原来匿名用户的 FTP 根目录所有者/组都是 root，所以匿名用户没有写入权限，那我们将所有者修改为 ftp 试试吧。

[root@linuxprobe ~]# chown  ftp /var/ftp/pub

此时再用 ftp 命令尝试登入 FTP 服务并创建文件：

ftp> mkdir files

550 Create directory operation failed.

可恶!又报错了!!虽然这次报错代码还是 550，但前面提示**权限拒绝**，这次是**操作失败**，马上想到是 SELinux 服务在捣鬼。

查看所有与 ftp 相关的 SELinux 规则：

[root@linuxprobe ~]# **getsebool -a | grep ftp**

```
  ftp_home_dir --> off
  ftpd_anon_write --> off
  ftpd_connect_all_unreserved --> off
  ftpd_connect_db --> off
  ftpd_full_access --> off
  ftpd_use_cifs --> off
  ftpd_use_fusefs --> off
  ftpd_use_nfs --> off
  ftpd_use_passive_mode --> off
  httpd_can_connect_ftp --> off
  httpd_enable_ftp_server --> off
  sftpd_anon_write --> off
  sftpd_enable_homedirs --> off
  sftpd_full_access --> off
  sftpd_write_ssh_home --> off
  tftp_anon_write --> off
tftp_home_dir --> off
```

设置 SELinux 服务对 ftp 服务的访问规则策略为允许。

[root@linuxprobe ~]# setsebool -P ftpd_full_access=on

此时再来创建文件或目录就没有问题了：

[root@linuxprobe ~]# **ftp 192.168.10.10**

Connected to 192.168.10.10 (192.168.10.10).

220 (vsFTPd 3.0.2)

Name (192.168.10.10:root): **anonymous**

331 Please specify the password.

Password:**敲击回车**

230 Login successful.

Remote system type is UNIX.

Using binary mode to transfer files.

ftp> **cd pub**

250 Directory successfully changed.

ftp> **mkdir files**

257 "/pub/files" created

ftp> **rename files database**

350 Ready for RNTO.

250 Rename successful.

ftp> **rmdir database**
250 Remove directory operation successful.
ftp> **exit**
221 Goodbye.

**请注意:当您完成本实验后请还原虚拟机快照再进行下一个实验，否则可能导致配置文件冲突而报错。**

### 10.3.2 本地用户模式

既然要使用本地用户模式，而本地用户模式确实要比匿名访问模式更加的安全，所以本实验中会关闭匿名访问模式。

vsftpd 服务程序默认已经允许本地用户模式，我们要做的是添加设置本地用户模式权限的参数：

[root@linuxprobe ~]# vim /etc/vsftpd/vsftpd.conf

| 参数 | 作用 |
| --- | --- |
| anonymous_enable=NO | 禁止匿名访问模式。 |
| local_enable=YES | 允许本地用户模式。 |
| write_enable=YES | 设置可写入权限。 |
| local_umask=022 | 本地用户模式创建文件的 umask 值。 |
| userlist_deny=YES | 参数值为 YES 即禁止名单中的用户，参数值为 NO 则代表仅允许名单中的用户。 |
| userlist_enable=YES | 允许“禁止登陆名单”，名单文件为 ftpusers 与 user_list。 |

确认填写正确后保存并退出 vsftpd.conf 文件，然后重启 vsftpd 服务程序并设置为开机自启动。

[root@linuxprobe ~]# *systemctl restart vsftpd*
[root@linuxprobe ~]# *systemctl enable vsftpd*
ln -s '/usr/lib/systemd/system/vsftpd.service' '/etc/systemd/system/multi-user.target.wants/vsftpd.service'

因为在红帽 RHCSA、RHCE 或 RHCA 考试后都要重启您的实验机再执行判分脚本。

所以请读者在日常工作中也要记得将需要的服务加入到开机启动项中:” **systemctl enable vsftpd** “。

如果重启 vsftpd 服务程序时没有报错，此时便可以使用 FTP 客户机(192.168.10.20)尝试登入 FTP 服务了~

我们先来看下 **ftpusers** 或 **user_list** 文件中禁止登陆用户名单：

**root**
bin
daemon
adm
lp
sync
shutdown
halt
mail
news
uucp
operator
games
nobody

vsftpd 服务为了让 FTP 服务更加的安全，默认禁止以 root 身份登入，那么创建个普通用户吧：

[root@linuxprobe ~]# useradd linuxprobe

为 linuxprobe 用户设置密码：

[root@linuxprobe ~]# **passwd linuxprobe**
Changing password for user linuxprobe.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.

**在客户端尝试登入 FTP 服务：**

[root@linuxprobe ~]# **ftp 192.168.10.10**
Connected to 192.168.10.10 (192.168.10.10).
220 (vsFTPd 3.0.2)
Name (192.168.10.10:root): **linuxprobe**
331 Please specify the password.
Password:**输入用户的本地密码**
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> **mkdir files**
550 Create directory operation failed.

有了上面配置匿名访问模式的经验，这次再遇到了"**操作被拒绝**"，应该马上想到 SELinux 了吧。

查看所有与 ftp 相关的 SELinux 规则：

[root@linuxprobe ~]# **getsebool -a | grep ftp**
ftp_home_dir --> off
ftpd_anon_write --> off
ftpd_connect_all_unreserved --> off
ftpd_connect_db --> off
ftpd_full_access --> off
ftpd_use_cifs --> off
ftpd_use_fusefs --> off
ftpd_use_nfs --> off
ftpd_use_passive_mode --> off
httpd_can_connect_ftp --> off
httpd_enable_ftp_server --> off
sftpd_anon_write --> off
sftpd_enable_homedirs --> off
sftpd_full_access --> off
sftpd_write_ssh_home --> off
tftp_anon_write --> off
tftp_home_dir --> off

设置 SELinux 对 FTP 服务的规则为允许：

[root@linuxprobe ~]# setsebool -P ftpd_full_access=on

此时再来创建文件或目录就没有问题了：

[root@linuxprobe ~]# **ftp 192.168.10.10**

Connected to 192.168.10.10 (192.168.10.10).

220 (vsFTPd 3.0.2)

Name (192.168.10.10:root): **linuxprobe**

331 Please specify the password.

Password:**输入用户本地密码**

230 Login successful.

Remote system type is UNIX.

Using binary mode to transfer files.

ftp> **mkdir files**

257 "/home/linuxprobe/files" created

ftp> **rename files database**

350 Ready for RNTO.

250 Rename successful.

ftp> **rmdir database**

250 Remove directory operation successful.

ftp> **exit**

221 Goodbye.

[/pre]

**请注意:当您完成本实验后请还原虚拟机快照再进行下一个实验，否则可能导致配置文件冲突而报错。**

### 10.3.3 虚拟用户模式

因为虚拟用户模式的帐号口令都不是真实系统中存在的，所以只要配置妥当虚拟用户模式会比本地用户模式更加安全，但是 Vsftpd 服务配置虚拟用户模式的操作步骤相对复杂一些，具体流程如下：

第 1 步:建立虚拟 FTP 用户数据库文件。

第 2 步:创建 FTP 根目录及虚拟用户映射的系统用户。

第 3 步:建立支持虚拟用户的 PAM 认证文件。

第 4 步:在 vsftpd.conf 文件中添加支持配置。

第 5 步:为虚拟用户设置不同的权限。

第 6 步:重启 vsftpd 服务，验证实验效果。

**第 1 步:建立虚拟 FTP 用户数据库文件。**

切换至 vsftpd 程序目录：

[root@linuxprobe ~]# cd /etc/vsftpd/

创建用于生成 FTP 用户数据库的原始帐号和密码文件：

[root@linuxprobe vsftpd]# **vim vuser.list**

//单数行为帐号，双数行为密码。

linuxprobe

pa33w0rd

blackshield

pa22w1rd

使用 db_load 命令用 HASH 算法生成 FTP 用户数据库文件 vuser.db：

[root@linuxprobe vsftpd]# db_load -T -t hash -f vuser.list vuser.db

查看数据库文件的类型：

[root@linuxprobe vsftpd]# file vuser.db

vuser.db: Berkeley DB (Hash, version 9, native byte-order)

FTP 用户数据库内容很敏感，所以权限给小一些：

[root@linuxprobe vsftpd]# chmod 600 vuser.db

删除原始的帐号和密码文件：

[root@linuxprobe vsftpd]# rm -f vuser.list

**第 2 步:创建 FTP 根目录及虚拟用户映射的系统用户。**

创建用户 virtual 并设置为不允许登陆系统并定义该用户的家目录：

[root@linuxprobe ~]# useradd -d /var/ftproot -s /sbin/nologin virtual

查看该用户的家目录权限：

[root@linuxprobe ~]# ls -ld /var/ftproot/

drwx------. 3 virtual virtual 74 Jul 14 17:50 /var/ftproot/

为保证其他用户可以访问，给予 rwxr-xr-x 权限：

[root@linuxprobe ~]# chmod -Rf 755 /var/ftproot/

**第 3 步:建立支持虚拟用户的 PAM 认证文件：**

[root@linuxprobe ~]# **vim /etc/pam.d/vsftpd.vu**

//参数 db 用于指向刚刚生成的 vuser.db 文件，但不要写后缀。

auth    required    pam_userdb.so db=/etc/vsftpd/vuser

account    required    pam_userdb.so db=/etc/vsftpd/vuser

**第 4 步:在 vsftpd.conf 文件中添加支持配置。**

既然要使用虚拟用户模式，而虚拟用户模式确实要比匿名访问模式更加的安全，配置的同时也关闭匿名开放模式。

[root@linuxprobe ~]# vim /etc/vsftpd/vsftpd.conf

| 参数 | 作用 |
|---|---|
| anonymous_enable=NO | 禁止匿名开放模式。 |
| local_enable=YES | 允许本地用户模式。 |
| guest_enable=YES | 开启虚拟用户模式。 |
| guest_username=virtual | 指定虚拟用户帐号。 |
| pam_service_name=vsftpd.vu | 指定 pam 文件。 |
| allow_writeable_chroot=YES | 允许禁锢的 FTP 根目录可写而不拒绝用户登入请求。 |

**第 5 步:为虚拟用户设置不同的权限**

现在不论是 linuxprobe 还是 blackshield 帐户，他们的权限都是相同的——默认不能上传、创建、修改文件，如果希望用户 blackshield 能够完全的管理 FTP 内的资料，就需要让 FTP 程序支持独立的用户权限配置文件了：

指定用户独立的权限配置文件存放的目录：

[root@linuxprobe ~]# vim /etc/vsftpd/vsftpd.conf

user_config_dir=/etc/vsftpd/vusers_dir

创建用户独立的权限配置文件存放的目录：

[root@linuxprobe ~]# mkdir /etc/vsftpd/vusers_dir/

切换进入到该目录中:
[root@linuxprobe ~]# cd /etc/vsftpd/vusers_dir/
创建空白的 linuxprobe 的配置文件:
[root@linuxprobe vusers_dir]# touch linuxprobe
指定 blackshield 用户的具体权限:
[root@linuxprobe vusers_dir]# **vim blackshield**
anon_upload_enable=YES
anon_mkdir_write_enable=YES
anon_other_write_enable=YES

**第 6 步:重启 vsftpd 服务，验证实验效果。**

确认填写正确后保存并退出 vsftpd.conf 文件，重启 vsftpd 程序并设置为开机后自动启用:
[root@linuxprobe ~]# *systemctl restart vsftpd*
[root@linuxprobe ~]# *systemctl enable vsftpd*
ln -s '/usr/lib/systemd/system/vsftpd.service' '/etc/systemd/system/multi-user.target.wants/vsftpd.service
因为在红帽 RHCSA、RHCE 或 RHCA 考试后都要重启您的实验机再执行判分脚本。
所以请读者在日常工作中也要记得将需要的服务加入到开机启动项中:"**systemctl enable vsftpd**"。
如果重启 vsftpd 并没有看到报错，此时可使用 FTP 客户机(192.168.10.20)尝试登入 FTP 服务了:
[root@linuxprobe ~]# **ftp 192.168.10.10**
Connected to 192.168.10.10 (192.168.10.10).
220 (vsFTPd 3.0.2)
Name (192.168.10.10:root): **blackshield**
331 Please specify the password.
Password:**此处输入虚拟用户的密码**
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> **mkdir files**
550 Create directory operation failed.
有了上面配置匿名访问模式和本地用户模式的经验，这次再遇到了"操作被拒绝"，应该马上想到 SELinux 服务了吧。
查看所有与 ftp 相关的 SELinux 规则:
[root@linuxprobe ~]# **getsebool -a | grep ftp**
设置 SELinux 对 FTP 服务的规则为允许:
[root@linuxprobe ~]# setsebool -P ftpd_full_access=on
此时再来创建文件或目录就没有问题了:
[root@linuxprobe ~]# **ftp 192.168.10.10**
Connected to 192.168.10.10 (192.168.10.10).
220 (vsFTPd 3.0.2)
Name (192.168.10.10:root): **blackshield**
331 Please specify the password.
Password:**此处输入虚拟用户的密码**
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> **mkdir files**

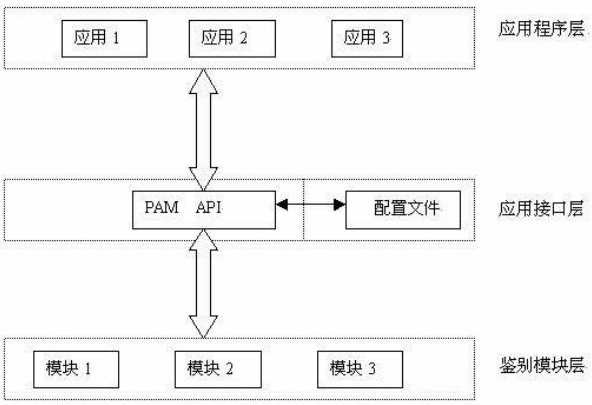
257 "/files" created
ftp> **rename files database**
350 Ready for RNTO.
250 Rename successful.
ftp> **rmdir database**
250 Remove directory operation successful.
ftp> **exit**
221 Goodbye.
使用 linuxprobe 用户创建（肯定会报错）：
[root@linuxprobe ~]# **ftp 192.168.10.10**
Connected to 192.168.10.10 (192.168.10.10).
220 (vsFTPd 3.0.2)
Name (192.168.10.10:root): **linuxprobe**
331 Please specify the password.
Password:**此处输入虚拟用户的密码**
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> **mkdir files**
550 Permission denied.
ftp> **exit**
221 Goodbye.
有没有觉得很有意思？当然读者们在工作中要学会灵活的搭配参数，不要完全按照实验操作而脱离客户需求。

**10.4 可插拔认证模块 PAM**

刚刚在上面 Vsftpd 服务的虚拟用户模式中提到到了一个叫做 PAM 的东西，现在为大家简单的介绍下 PAM。

可插拔认证模块 PAM(Pluggable Authentication Modules)是一种认证机制，通过一些动态链接库和统一的 API 将系统提供的服务与认证方式分开，使得系统管理员可以根据需求灵活的调整服务程序的不同认证方式。

通俗来讲 PAM 是一组安全机制的模块(插件)，让系统管理员可以轻易的调整服务程序的认证方式，**此时可以不必对应用程序做任何的修改**，易用性很强，PAM 采取了分层设计的思想——应用程序层、应用接口层、鉴别模块层。




PAM API 作为应用程序层与鉴别模块层的连接纽带，让应用程序可以根据需求灵活的在其中插入所需的鉴别功能模块，当应用程序需要 PAM 认证时，一般在应用程序中定义负责其认证的 PAM 配置文件，真正灵活的实现了认证功能，读者不必精通 PAM 模块，也不用对参数做细致的讲解，只需认识 PAM 模块的重要目录：

**/lib/security**:pam 认证模块。

**/etc/pam.d:**针对不同服务而定义好的 pam 配置文件。

例如 vsftpd 程序就会在其主配置文件(“**/etc/vsftpd/vsftpd.conf**“)中写入下面的参数:

pam_service_name=vsftpd

表示登陆 FTP 服务器时是根据/etc/pam.d/vsftpd 的文件内容进行安全认证的。

因为我们平时不会经常修改 PAM 配置文件，而且 PAM 模块相对比较复杂，所以不在本章中继续讲解，读者必需能够理解刚刚实验中出现的 vsftpd.vu 文件的作用以及存放位置。

# 第 11 章 使用 Samba 或 NFS 实现文件共享。

**章节简述:**
本章节为读者讲述文件共享系统的作用，了解 Samba 与 NFS 服务程序的开发背景以及用法。
详细逐条讲解 Samba 服务配置参数，演示安全共享文件的配置策方法，并使用 autofs 服务程序自动挂载设备，
学会后即可实现 Linux 系统之间或与 Windows 系统之间的文件共享，以及在共享文件时如何配置防火墙与 SELinux 策略规则。

## 11.1 了解文件共享服务

早期网络想要在不同主机之间共享文件大多要用 FTP 协议来传输，但 FTP 协议仅能做到传输文件却不能直接修改对方主机的资料数据，这样确实不太方便，于是便出现了 NFS 开源文件共享程序:NFS(NetworkFile System)是一个能够将多台 Linux 的远程主机数据挂载到本地目录的服务，**属于轻量级的文件共享服务，不支持 Linux 与 Windows 系统间的文件共享。**

随后在 1991 年时大学生 Tridgwell 为了解决 Linux 与 Windows 系统之间共享文件的问题，便开发出了 SMB 协议与 Samba 服务程序。
SMB(Server Messages Block)协议:实现局域网内文件或打印机等资源共享服务的协议。
当时 Tridgwell 想要注册 SMBServer 这个商标，但却被因为 SMB 是没有意义的字符被拒绝了，经过 Tridgwell 不断翻看词典，终于找到了一个拉丁舞蹈的名字——SAMBA，而这个热情舞蹈的名字中又恰好包含了 SMB(SAMBA)，于是这便是 Samba 程序名字的由来。
Samba 服务程序是一款基于 SMB 协议并由服务端和客户端组成的开源文件共享软件，实现了 Linux 与 Windows 系统间的文件共享。

## 11.2 Samba 服务

### 11.2.1 安装服务程序

通过安装 Samba 服务程序后细致的分析其配置文件参数，更能够帮助读者们理解 Samba 服务的安全验证方式。

**安装 Samba 服务软件包:**

```
[root@linuxprobe Desktop]# yum install samba
Loaded plugins: langpacks, product-id, subscription-manager
………………省略部分安装过程………………
Installing:
 samba            x86_64          4.1.1-31.el7           rhel7          527 k
………………省略部分安装过程………………
Complete!
```

**浏览 Samba 配置文件:**

```
[root@linuxprobe ~]# cat/etc/samba/smb.conf
```

配置文件竟然有 320 行！有没有被吓到？其实 Samba 服务配置文件中大部分是注释信息，我们可以来筛选过滤下:

**备份原始的配置文件:**

```
[root@linuxprobe ~]# mv /etc/samba/smb.conf /etc/samba/smb.conf.bak
```

**过滤掉无用的内容:**

先使用 cat 命令读入 Smb 配置文件后通过 grep 命令-v 参数（反向选择）去掉所有注释信息，然后分别删选掉包含#号的行（“#”），包含;号的行（“;”）以及所有的空白行（“^$”），最后最后将过滤后的信息覆盖写入到/etc/samba/smb.conf 文件中。

cat /etc/samba/smb.conf.bak | grep -v "#" | grep -v ";" | grep -v "^$" > /etc/samba/smb.conf

**让我们来看看过滤后的配置文件吧:**

| [global] | | #全局参数。 |
|---|---|---|
| | workgroup = MYGROUP | #工作组名称。 |
| | server string = Samba Server Version %v | #服务器介绍信息,参数%v 为显示 SMB 版本号。 |
| | log file = /var/log/samba/log.%m | #定义日志文件存放位置与名称，参数%m 为来访的主机名。 |
| | max log size = 50 | #定义日志文件最大容量为 50Kb。 |
| | security = user | #安全验证的方式,总共有 4 种。 |
| | #share:来访主机无需验证口令，更加方便，但安全性很差。 | |
| | #user:需由 SMB 服务验证来访主机提供的口令后才可建立访问,更加的安全。 | |
| | #server:使用独立的远程主机验证来访主机提供的口令（集中管理帐号)。 | |
| | #domain:使用 PDC 来完成验证 | |
| | passdb backend = tdbsam | #定义用户后台的类型，共有 3 种。 |
| | #smbpasswd:使用 SMB 服务的 smbpasswd 命令给系统用户设置 SMB 密码。 | |
| | #tdbsam:创建数据库文件并使用 pdbedit 建立 SMB 独立的用户。 | |
| | #ldapsam:基于 LDAP 服务进行帐户验证。 | |
| | load printers = yes | #设置是否当 Samba 服务启动时共享打印机设备。 |
| | cups options = raw | #打印机的选项 |
| [homes] | | #共享参数 |
| | comment = Home Directories | #描述信息 |
| | browseable = no | #指定共享是否在“网上邻居”中可见。 |
| | writable = yes | #定义是否可写入操作，与"read only"相反。 |
| [printers] | | #打印机共享参数 |

| | | |
|---|---|---|
| | comment = All Printers | |
| | path = /var/spool/samba | #共享文件的实际路径(重要)。 |
| | browseable = no | |
| | guest ok = no | #是否所有人可见，等同于"public"参数。 |
| | writable = no | |
| | printable = yes | |

**标准的 Samba 共享参数是这样的：**

| 参数 | 作用 |
|---|---|
| [linuxprobe] | 共享名称为 linuxprobe |
| comment = Do not arbitrarily modify the database file | 警告用户不要随意修改数据库 |
| path = /home/database | 共享文件夹在/home/database |
| public = no | 关闭所有人可见 |
| writable = yes | 允许写入操作 |

我们将上面的配置参数直接追加到 SMB 服务配置文件(/etc/samba/smb.conf)并重启 SMB 服务程序即可生效。
但此时 SMB 服务默认的验证模式为 user，我们需要先创建用户数据库后才可以正常使用，现在来学习下如何创建吧~

### 11.2.2 安全共享文件

使用 Samba 服务口令验证方式可以让共享文件更加的安全，做到仅让信任的用户访问，而且验证过程也很简单，要想使用口令验证模式，我们需要先需要创建 Samba 服务独立的数据库。

**第 1 步:检查当前是否为 user 验证模式。**

[root@linuxprobe ~]# cat /etc/samba/smb.conf

```
security = user
passdb backend = tdbsam
```

**第 2 步:创建共享文件夹：**

[root@linuxprobe ~]# mkdir /database

**第 3 步:描述共享文件夹信息。**

在 **SMB 服务主配置文件**的最下面追加共享文件夹的配置参数：

**[database]**

comment = Do not arbitrarily modify the database file

path = /database

public = no

writable = yes

保存 **smb.conf** 文件后重启启动 SMB 服务：

[root@linuxprobe ~]# systemctl restart smb

添加到开机启动项：

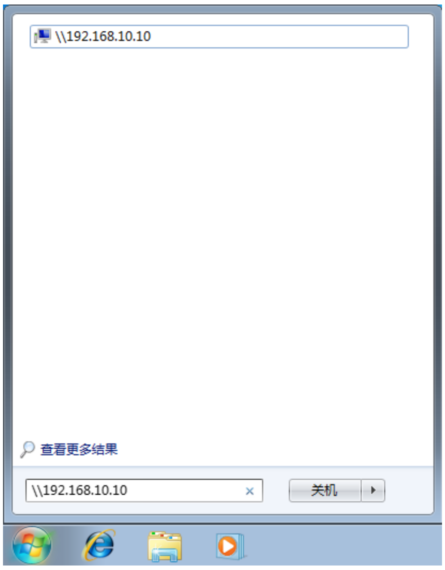
[root@linuxprobe ~]# *systemctl enable smb*

ln -s '/usr/lib/systemd/system/smb.service' '/etc/systemd/system/multi-user.target.wants/smb.service'

**第 4 步：使用 Windows 主机尝试访问**

读者按照下表的 IP 地址规划动手配置下 Windows 的网卡参数，应该都会吧~

| 主机名称 | 操作系统 | IP 地址 |
|---|---|---|
| Samba 共享服务器 | 红帽 RHEL7 操作系统 | 192.168.10.10 |
| 客户端 | 红帽 RHEL7 操作系统 | 192.168.10.20 |
| 客户端 | 微软 Windows7 操作系统 | 192.168.10.30 |




**在 Windows 主机的运行框中输入远程主机的信息**




**此时访问 Samba 服务报错**

此时访问 Samba 服务是报错的，如果读者已经看完 Apache(httpd)服务程序的章节，应该还记得防火墙和 SELinux 规则吧。

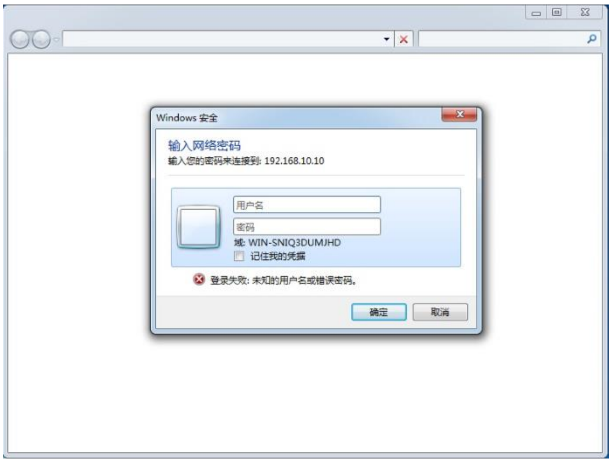
**第 5 步:清空防火墙规则链：**

Windows 访问 Samba 主机提示报错，我们怀疑是 Iptables 阻止了访问操作，于是执行：

[root@linuxprobe ~]# Iptables -F

[root@linuxprobe ~]# service iptables save

因为 Windows 系统的缓存关系，可能需要先**重启**下 **Windows 主机**再尝试访问 Samba 共享。




**Windows 系统被要求验证帐户口令**

那么这个问题就是出在 Iptables 防火墙的默认规则中了，所以请对 SELinux 多一点耐心，不要直接关闭 SELinux。

**第 6 步:创建 SMB 服务独立的帐号。**

现在 Windows 系统要求先验证后才能访问共享，而 SMB 服务配置文件中密码数据库后台类型为”**tdbsam**“，所以这个帐户和口令是 Samba 服务的独立帐号信息，我们需要使用 **pdbedit** 命令来创建 SMB 服务的用户数据库。

**pdbedit 命令用于管理 SMB 服务的帐户信息数据库，格式为：“pdbedit [选项] 帐户”。**

| 参数 | 作用 |
| --- | --- |
| -a 用户名 | 建立 Samba 用户 |
| -x 用户名 | 删除 Samba 用户 |
| -L | 列出用户列表 |
| -Lv | 列出用户详细信息的列表 |

创建系统用户：

[root@linuxprobe ~]# useradd smbuser

将此系统用户提升为 SMB 用户：

[root@linuxprobe ~]# **pdbedit -a -u smbuser**

new password:设置 SMB 服务独立的密码

retype new password:

Unix username: smbuser

NT username:

Account Flags: [U ]

User SID: S-1-5-21-4146456071-3435711857-2069708454-1000

Primary Group SID: S-1-5-21-4146456071-3435711857-2069708454-513

Full Name:

Home Directory: \\linuxprobe\smbuser

HomeDir Drive:

Logon Script:

Profile Path: \\linuxprobe\smbuser\profile

Domain: LINUXPROBE

Account desc:

Workstations:

Munged dial:

Logon time: 0

Logoff time: Wed, 06 Feb 2036 23:06:39 CST

Kickoff time: Wed, 06 Feb 2036 23:06:39 CST

Password last set: Sat, 11 Jul 2015 18:27:04 CST

Password can change: Sat, 11 Jul 2015 18:27:04 CST

Password must change: never

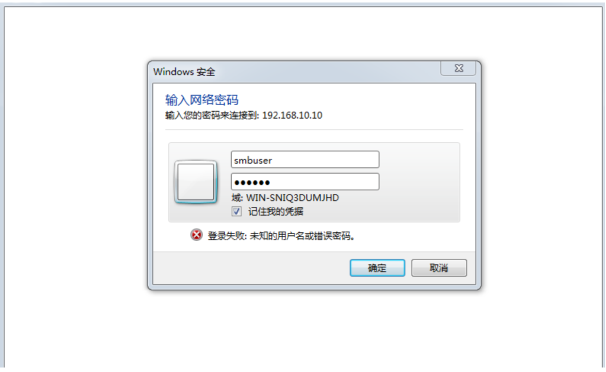
Last bad password : 0

Bad password count : 0

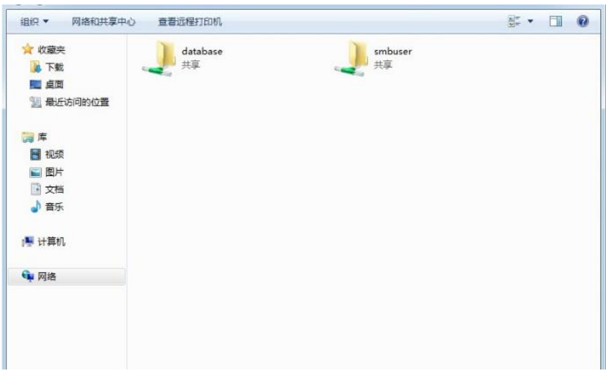
Logon hours : FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

**第 7 步:使用 Windows 主机验证共享结果:**

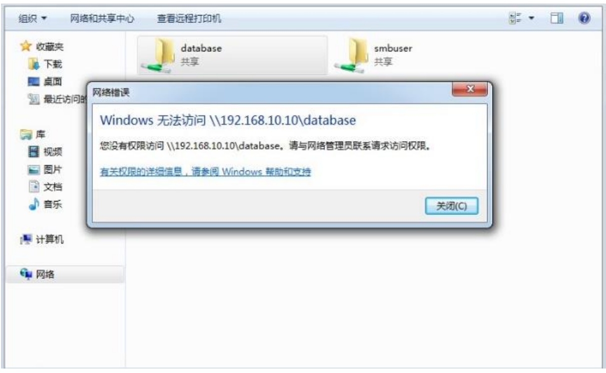
Windows 验证 SMB 服务口令




Windows 成功访问 SMB 服务




Windows 进入共享目录失败




**第 8 步：允许 SELinux 规则**

使用 Windows 主机访问 Samba 共享果然可以使用 smbuser 用户登入，但对于共享文件这么重要的事情，SELinux 一定会强制管理，刚刚没有妥当的配置好 SELinux，现在果然又报错了。

将共享目录的所有者和所有组设置为 smbuser 用户：

[root@linuxprobe ~]# chown -Rf smbuser:smbuser /database

允许 SELinux 对于 SMB 用户共享家目录的布尔值：

[root@linuxprobe ~]# setsebool -P samba_enable_home_dirs on

将共享目录的 SELinux 安全上下文设置妥当：

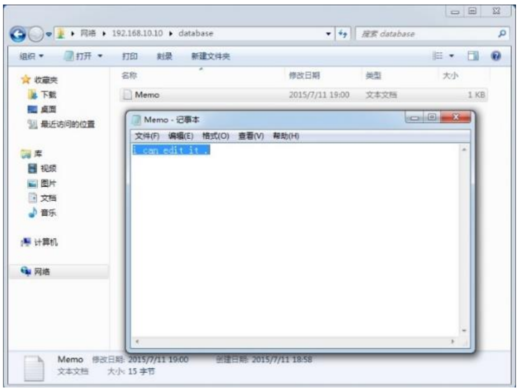
[root@linuxprobe ~]# semanage fcontext -a -t samba_share_t /database

使新的安全上下文立即生效：

[root@linuxprobe ~]# restorecon -Rv /database/

**第 9 步：使用 Windows 主机验证共享结果**

我们配置好 Samba 服务后又陆续的调整好了 Iptables 防火墙与 SELinux 安全规则，现在终于可以正常的使用共享了。



**使用 SMB 服务并创建文件**

**第 10 步：使用 Linux 主机验证共享结果**

刚刚好像让读者产生了一些小误解，Samba 服务程序并不仅仅是能够实现 Linux 与 Windows 系统间的文件共享，还可以实现 Linux 系统之间的文件共享哦，先动手配置下客户端主机的 IP 地址吧：

| 主机名称 | 操作系统 | IP 地址 |
| --- | --- | --- |
| Samba 共享服务器 | 红帽 RHEL7 操作系统 | 192.168.10.10 |
| 客户端 | 红帽 RHEL7 操作系统 | 192.168.10.20 |
| 客户端 | 微软 Windows7 操作系统 | 192.168.10.30 |

在客户端安装 cifs-utils 软件包：

```
[root@linuxprobe ~]# yum install -y cifs-utils
Loaded plugins: langpacks, product-id, subscription-manager
………………省略部分安装过程………………
Installing:
 cifs-utils        x86_64        6.2-6.el7        rhel7        83 k
………………省略部分安装过程………………
Complete!
```

创建挂载目录：

```
[root@linuxprobe ~]# mkdir /database
```

在 root 家目录创建认证文件(依次为 SMB 用户名、SMB 用户密码、SMB 共享域)：

```
[root@linuxprobe ~]# vim auth.smb
username=smbuser
password=redhat
domain=MYGROUP
```

此文件太重要了，权限应该给小一些：

```
[root@linuxprobe ~]# chmod -Rf 600 auth.smb
```

配置其挂载信息（内容依次为远程共享信息、本地挂载目录、文件系统类型、认证文件以及开机自检选项)：

```
[root@linuxprobe ~]# vim /etc/fstab
//192.168.10.10/database /database cifs credentials=/root/auth.smb 0 0
```

使用 mount 命令的-a 参数挂载所有在 fstab 文件中定义的文件信息：

[root@linuxprobe ~]# mount -a

成功挂载 Samba 的共享目录（能够看到共享文件了）：

[root@linuxprobe ~]# cat /database/Memo.txt

i can edit it .

Samba 服务真的是太强大了，不仅能够实现 Linux 系统之间分享数据还能与 Windows 主机进行文件共享。

**11.3 NFS 网络文件系统**

NFS(Network Files System)即网络文件系统，NFS 文件系统协议允许网络中的主机通过 TCP/IP 协议进行资源共享，NFS 客户端可以像使用本地资源一样读写远端 NFS 服务端的资料，需要注意 NFS 服务依赖于 RPC 服务与外部通信，所以必需保证 RPC 服务能够正常注册服务的端口信息才能正常使用 NFS 服务。

有个学员问过 NFS 是不是 need for speed 的缩写啊？哈哈，NFS 配置和使用都是非常快捷，所以这么说也是有道理的。

**红帽 RHEL7 系统已经默认安装 NFS 服务：**

[root@linuxprobe ~]# **yum install nfs-utils**

Loaded plugins: langpacks, product-id, subscription-manager

(1/2): rhel7/group_gz | 134 kB 00:00

(2/2): rhel7/primary_db | 3.4 MB 00:00

**Package 1:nfs-utils-1.3.0-0.el7.x86_64 already installed and latest version**

Nothing to do

本次的实验需要两台 Linux 主机，网络配置情况：

| 主机名称 | 操作系统 | IP 地址 |
| --- | --- | --- |
| NFS 服务端 | 红帽 RHEL7 操作系统 | 192.168.10.10 |
| NFS 客户端 | 红帽 RHEL7 操作系统 | 192.168.10.20 |

**第 1 步:创建 NFS 服务端的共享目录。**

清空 iptables 默认的规则链：

[root@linuxprobe ~]# iptables -F

保存清空后的 iptables 规则：

[root@linuxprobe ~]# service iptables save

创建 nfsfile 共享目录：

[root@linuxprobe ~]# mkdir /nfsfile

写入一个文件，用于 NFS 客户端读取：

[root@linuxprobe ~]# echo "welcome to linuxprobe.com" > /nfsfile/readme

NFS 服务端配置文件是”/etc/exports”，用于定义要共享的目录以及相应权限。

[root@linuxprobe ~]# vim /etc/exports

//格式为:共享目录的绝对路径 允许访问 NFS 资源的客户端(权限参数)

**/nfsfile 192.168.10.* (rw,sync,root_squash)**

NFS 配置共享的参数有：

| 参数 | 作用 |
| --- | --- |
| ro | 只读默认 |
| rw | 读写模式 |
| root_squash | 当 NFS 客户端使用 root 用户访问时，映射为 NFS 服务端的匿名用户。 |

| | |
|---|---|
| no_root_squash | 当 NFS 客户端使用 root 用户访问时，映射为 NFS 服务端的 root 用户。 |
| all_squash | 不论 NFS 客户端使用任何帐户，均映射为 NFS 服务端的匿名用户。 |
| sync | 同时将数据写入到内存与硬盘中，保证不丢失数据。 |
| async | 优先将数据保存到内存，然后再写入硬盘，效率更高，但可能造成数据丢失。 |

**第 2 步:启动 NFS 服务端**

刚刚讲到 NFS 服务是依赖于 RPC 服务的，但在红帽 RHEL7 系统中 RPC 服务已经默认运行(active)了，所以无需再配置 RPC 服务啦。

[root@linuxprobe ~]# systemctl status rpcbind

启动 nfs-server 程序：

[root@linuxprobe ~]# systemctl start nfs-server

设置 NFS 服务端为开机启动：

[root@linuxprobe ~]# systemctl enable nfs-server

**第 3 步：配置 NFS 客户端**

如果 NFS 客户端也是红帽 RHEL7 系统，那么软件包 **nfs-utils** 一定也是已经默认安装，直接挂载共享就可以了。

**showmount 命令用于查询 NFS 服务端共享信息，格式为：“showmount [参数] [远程主机]”。**

| 参数 | 作用 |
|---|---|
| -e | 显示 NFS 服务端的共享列表 |
| -a | 显示本机挂载 NFS 资源的情况 |
| -v | 显示版本号 |

查询远程 NFS 服务端中可用的共享资源：

[root@linuxprobe ~]# showmount -e 192.168.10.10
Export list for 192.168.10.10:
/nfsfile (everyone)

创建本地挂载目录：

[root@linuxprobe ~]# mkdir /nfsfile
[root@linuxprobe ~]# mount -t nfs 192.168.10.10:/nfsfile /nfsfile

顺利查看到刚刚写入文件内容：

[root@linuxprobe ~]# cat /nfsfile/readme
welcome to linuxprobe.com

如果希望开机后自动将 NFS 资源挂载到本地，那么就可以通过修改 fstab 文件来实现：

[root@linuxprobe ~]# vim /etc/fstab
192.168.10.10:/nfsfile /nfsfile nfs defaults 0 0

**11.4 AutoFs 自动挂载服务**

AutoFs 服务与 Mount/Umount 命令不同之处在于它是一种守护进程，只有检测到用户试图访问一个尚未挂载的文件系统时才自动的检测并挂载该文件系统，换句话说，将挂载信息填入**/etc/fstab** 文件后系统将在每次开机时都自动将其挂载，而运行 AutoFs 后则是当用户需要使用该文件系统了才会动态的挂载，节约网络与系统资源。

**模拟训练:每次进入/media/iso 目录时都会自动挂载镜像。**

| 主机名称 | 操作系统 | IP 地址 |
| --- | --- | --- |
| NFS 服务端 | 红帽 RHEL7 操作系统 | 192.168.10.10 |
| NFS 客户端 | 红帽 RHEL7 操作系统 | 192.168.10.20 |

安装 autofs 服务：

[root@linuxprobe ~]# yum install autofs
………………省略部分安装过程………………
Installing:
autofs x86_64 1:5.0.7-40.el7 rhel 550 k
Installing for dependencies:
hesiod x86_64 3.2.1-3.el7 rhel 30 k
………………省略部分安装过程………………
Complete!

启动 autofs 服务并加入到开机启动项中：

[root@linuxprobe ~]# *systemctl start autofs*
[root@linuxprobe ~]# *systemctl enable autofs*
ln -s '/usr/lib/systemd/system/autofs.service' '/etc/systemd/system/multi-user.target.wants/autofs.service'

修改 autofs 主配置文件（格式为：挂载目录 映射配置文件）：

[root@linuxprobe ~]# vim /etc/auto.master
/media /etc/iso.misc

编辑挂载配置参数文件（映射文件）：

[root@linuxprobe ~]# vim /etc/auto.misc
iso -fstype=iso9660,ro,nosuid,nodev :/dev/cdrom

重新加载启动 autofs 服务：

[root@linuxprobe ~]# systemctl restart autofs

进入自动挂载目录中：

[root@linuxprobe ~]# cd /media/iso

查看当前目录下的文件：

[root@linuxprobe iso]# **ls**
addons images Packages RPM-GPG-KEY-redhat-release
EFI isolinux release-notes TRANS.TBL
EULA LiveOS repodata
GPL media.repo RPM-GPG-KEY-redhat-beta

**本章结束，您可以在此写下笔记：**

# 第 12 章 使用 Bind 提供域名解析服务。

**章节简述:**

本章节将让您理解 DNS 服务程序的原理，学习正向解析与反向解析实验，掌握 DNS 主服务器、从服务器、缓存服务器的部署方法。够熟练配置区域信息文件与区域数据文件，以及通过使用分离解析技术让不同来源的用户得到更合适的解析结果。

DNS 服务作为互联网的基础设施，我们还可以配置 BIND 服务程序支持 TSIG 安全加密传输机制，从而保障解析数据不被嗅探监听。

## 12.1 了解域名解析服务

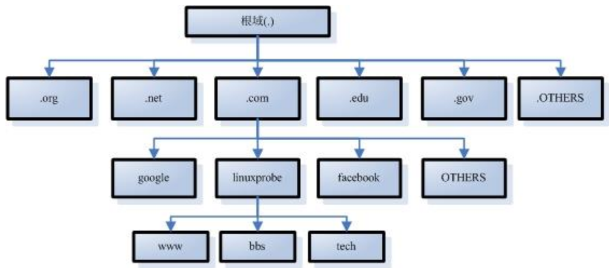
一般来讲域名比 IP 地址更加的有含义、也更容易记住，所以通常用户更习惯输入域名来访问网络中的资源，但是计算机主机在互联网中只能通过 IP 识别对方主机，那么就需要 DNS 域名解析服务了。

DNS 域名解析服务(Domain Name System)是用于解析域名与 IP 地址对应关系的服务，功能上可以实现**正向解析**与**反向解析**:

**正向解析**:根据主机名(域名)查找对应的 IP 地址。

**反向解析**:根据 IP 地址查找对应的主机名(域名)。

DNS 服务协议采用类似目录树的层次结构记录域名与 IP 地址的映射对应关系，形成一个分布式的数据库系统：[DNS 结构模型]




而单靠几台 DNS 服务器肯定不能满足全球如此多用户的需求，所以从工作形式上又分**主服务器**、**从服务器**和**缓存服务器**。

**主服务器**:在特定区域内具有唯一性、负责维护该区域内的域名与 IP 地址对应关系。

**从服务器**:从主服务器中获得域名与 IP 地址对应关系并维护，以防主服务器宕机等情况。

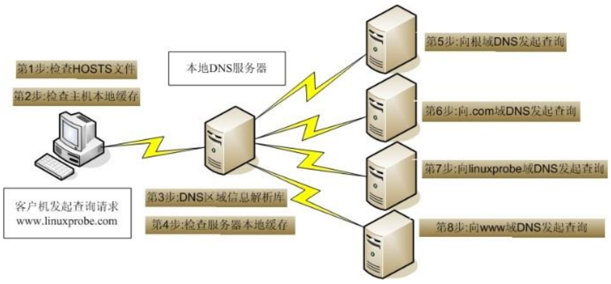
**缓存服务器**:通过向其他域名解析服务器查询获得域名与 IP 地址对应关系，提高重复查询时的效率。

这里还需要简单了解下，DNS 查询时还会分为**递归查询**与**迭代查询**。

**递归查询**:用于客户机向 DNS 服务器查询。

**迭代查询**:用于 DNS 服务器向其它 DNS 服务器查询。

**互联网 DNS 访问模型**——采用分布式数据结构保存海量区域数据信息，用户从互联网中查询过程大致流程为：DNS 查询流程图




## 12.2 安装 Bind 服务程序

伯克利互联网域名服务 BIND(Berkeley Internet Name Daemon)是一款全球互联网使用最广泛的能够提供安全可靠、快捷高效的域名解析服务程序。并且 Bind 服务程序还支持 chroot（change root）监牢安全机制，chroot 机制会限制 bind 服务程序仅能对自身配置文件进行操作，从而保证了整个服务器的安全，让既然如此，那么我们也就选择安装 bind-chroot 吧：

[root@linuxprobe ~]# yum install bind-chroot

………………省略部分安装过程………………

Complete!

域名解析服务 Bind 的程序名称叫做 **named**，服务程序的配置文件如下：

| 主程序 | /usr/sbin/named |
| --- | --- |
| 主配置文件 | /etc/named.conf |
| 区域配置文件 | /etc/named.rfc1912.zones |

那么先来分析下主配置文件吧(**有两处需要修改，请跟着一起操作**.):

```
[root@linuxprobe ~]# vim /etc/named.conf
// Provided by Red Hat bind package to configure the ISC BIND named(8) DNS
// server as a caching only nameserver (as a localhost DNS resolver only).
// See /usr/share/doc/bind*/sample/ for example named configuration files.
options {
//将下行中的 127.0.0.1 修改为 any，代表允许监听任何 IP 地址。
listen-on port 53 { 127.0.0.1; };
listen-on-v6 port 53 { ::1; };
directory "/var/named";
dump-file "/var/named/data/cache_dump.db";
statistics-file "/var/named/data/named_stats.txt";
memstatistics-file "/var/named/data/named_mem_stats.txt";
//将下行中的 localhost 修改为 any，代表允许任何主机查询。
allow-query { localhost; };
recursion yes;
dnssec-enable yes;
dnssec-validation yes;
dnssec-lookaside auto;
/* Path to ISC DLV key */
bindkeys-file "/etc/named.iscdlv.key";
managed-keys-directory "/var/named/dynamic";
pid-file "/run/named/named.pid";
session-keyfile "/run/named/session.key";
};
logging {
channel default_debug {
file "data/named.run";
severity dynamic;
};
};

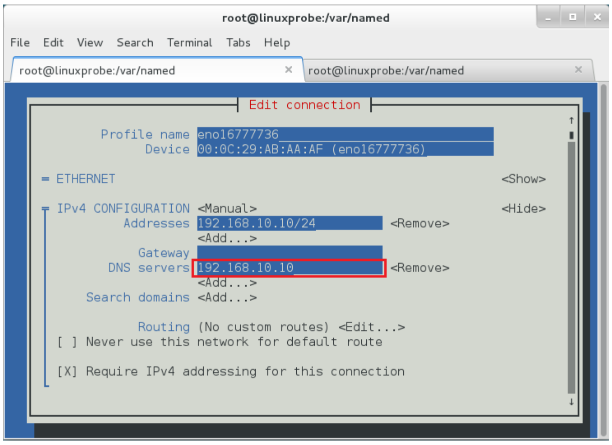
//此文件内定义了全球 13 台根 DNS 服务器的 IP 地址
zone "." IN {
type hint;
file "named.ca";
};
```

//此文件保存着正向与反向解析的区域信息，非常的重要。
include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";
当用户访问一个域名时(不考虑 hosts 文件等因素)，正常情况会向指定的 DNS 主机发送递归查询请求，如果该 DNS 主机中没有该域名的解析信息那么会不断向上级 DNS 主机进行迭代查询，其中最高等级(权威)的根 DNS 主机有 13 台，分别为:
根 DNS 服务器 IP 地址文件:/var/named/named.ca

| 名称 | 管理单位 | 地理位置 | IP 地址 |
|---|---|---|---|
| A | INTERNIC.NET | 美国-弗吉尼亚州 | 198.41.0.4 |
| B | 美国信息科学研究所 | 美国-加利弗尼亚州 | 128.9.0.107 |
| C | PSINet 公司 | 美国-弗吉尼亚州 | 192.33.4.12 |
| D | 马里兰大学 | 美国-马里兰州 | 128.8.10.90 |
| E | 美国航空航天管理局 | 美国加利弗尼亚州 | 192.203.230.10 |
| F | 因特网软件联盟 | 美国加利弗尼亚州 | 192.5.5.241 |
| G | 美国国防部网络信息中心 | 美国弗吉尼亚州 | 192.112.36.4 |
| H | 美国陆军研究所 | 美国-马里兰州 | 128.63.2.53 |
| I | Autonomica 公司 | 瑞典-斯德哥尔摩 | 192.36.148.17 |
| J | VeriSign 公司 | 美国-弗吉尼亚州 | 192.58.128.30 |
| K | RIPE NCC | 英国-伦敦 | 193.0.14.129 |
| L | IANA | 美国-弗吉尼亚州 | 199.7.83.42 |
| M • | WIDE Project | 日本-东京 | 202.12.27.33 |

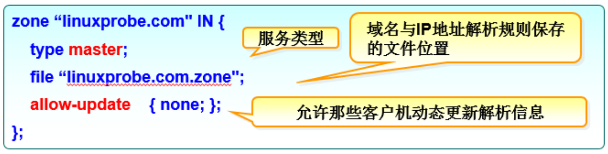
## 12.3 DNS 服务的解析实验

既然要开始搭建使用 DNS 服务程序啦，那么请将系统的 DNS 地址修改为本机，这样才能看到实验效果哦。



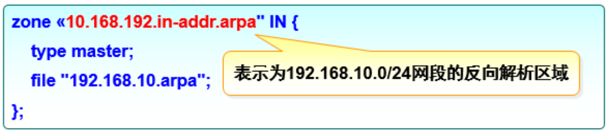
为了避免经常修改主配置文件 named.conf 而导致 DNS 服务出错，所以规则的区域信息保存在了”/etc/named.rfc1912.zones“文件中，这个文件用于定义域名与 IP 地址解析规则保存的文件位置以及区域服务类型等内容，一定要谨慎修改。

**正向解析区域文件格式:**




服务类型可以有三种:hint(根区域)、master(主区域)、slave(辅助区域)。

**反向解析区域文件格式:**




zone 区域中 IP 信息必需反写(如上图演示)，并且后面要写上”in-addr.arpa“。

接下来的实验中会分别对主配置文件、区域信息文件与区域数据文件做修改，当怀疑因配置参数而出错时可执行 **named-checkconf** 或 **named-checkzone** 命令来分别用于检查主配置与区域数据文件中语法或参数的错误。

### 12.3.1 正向解析实验

**第 1 步:配置区域数据信息。**

正向解析的作用是根据主机名(域名)查找到对应的 IP 地址，区域文件中已有一些默认信息，可不必理会，直接在下面追加即可:

```
[root@linuxprobe ~]# vim /etc/named.rfc1912.zones
    zone "linuxprobe.com" IN {
    type master;
    file "linuxprobe.com.zone";
    allow-update {none;};
    }
```

**第 2 步：配置解析数据信息：**

我们可以直接复制正向解析模板文件:”**/var/named/named.localhost**“，填写信息后即可直接使用。

切换工作目录到 bind(named)数据目录：

```
[root@linuxprobe ~]# cd /var/named/
```

查看区域数据文件的权限：

```
[root@linuxprobe named]# ls -al named.localhost
-rw-r-----. 1 root named 152 Jun 21 2007 named.localhost
```

执行 cp 命令时加入-a，代表连通复制原来文件的属性、所有者、组等信息：

```
[root@linuxprobe named]# cp -a named.localhost linuxprobe.com.zone
```

编辑 linuxprobe.com 域名的区域数据文件：

```
[root@linuxprobe named]# vim linuxprobe.com.zone
```

重启 named 服务让配置文件立即生效:

```
[root@linuxprobe named]# systemctl restart named
```

| $TTL 1D | #生存周期为 1 天 | | | |
|---|---|---|---|---|
| @ | IN SOA | linuxprobe.com. | root.linuxprobe.com. | ( |
| | #授权信息开始: | #DNS 区域的地址 | #域名管理员的邮箱(不要用@符号) | |

| | | | | |
|---|---|---|---|---|
| | | | 0;serial | #更新序列号 |
| | | | 1D;refresh | #更新时间 |
| | | | 1H;retry | #重试延时 |
| | | | 1W;expire | #失效时间 |
| | | | 3H;minimum | #无效解析记录的缓存时间 |
| | NS | ns.linuxprobe.com. | #域名服务器记录 | |
| ns | IN A | 192.168.10.10 | #地址记录(ns.linuxprobe.com.) | |
| | IN MX 10 | mail.linuxprobe.com. | #邮箱交换记录 | |
| mail | IN A | 192.168.10.10 | #地址记录(mail.linuxprobe.com.) | |
| www | IN A | 192.168.10.10 | #地址记录(www.linuxprobe.com.) | |
| bbs | IN A | 192.168.10.20 | #地址记录(bbs.linuxprobe.com.) | |

**第 3 步:检验解析结果。**

**nslookup** 命令用于检测能否从网络 DNS 服务器中查询到域名与 IP 地址的解析记录，检测 named 服务的解析能否成功。

此为查询 DNS 服务器的信息:

[root@linuxprobe ~]# nslookup
> www.linuxprobe.com
Server: 127.0.0.1
Address: 127.0.0.1#53

顺利的将域名的对应 IP 地址解析出来了:

Name: www.linuxprobe.com
Address: 192.168.10.10
> bbs.linuxprobe.com
Server: 127.0.0.1
Address: 127.0.0.1#53
Name: bbs.linuxprobe.com
Address: 192.168.10.20
> mail.linuxprobe.com
Server: 127.0.0.1
Address: 127.0.0.1#53
Name: mail.linuxprobe.com
Address: 192.168.10.10
> ns.linuxprobe.com
Server: 127.0.0.1
Address: 127.0.0.1#53

Name: ns.linuxprobe.com
Address: 192.168.10.10

### 12.3.2 反向解析实验

反向解析的作用是根据 IP 地址查找到对应的主机名（域名），在区域文件(named.rfc1912.zones)中默认已存在一些注释内容与区域信息，可不需要删除上面实验及默认区域信息，直接在下面追加即可。

**第 1 步：配置区域数据信息。**

[root@linuxprobe ~]# vim /etc/named.rfc1912.zones
zone "10.168.192.in-addr.arpa" IN {
type master;
file "192.168.10.arpa";
};

**第 2 步：配置解析数据信息。**

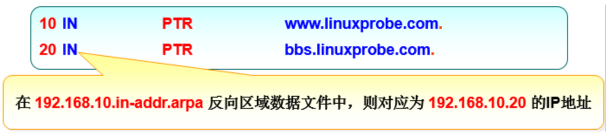
反向解析数据文件模版为:"/var/named/named.loopback"，我们可复制并填写信息后即可直接使用：

[root@linuxprobe named]# cp -a named.loopback 192.168.10.arpa

编辑 192.168.10.0/24 网段的数据文件：

[root@linuxprobe named]# vim 192.168.10.arpa

| $TTL 1D | | | | |
|---|---|---|---|---|
| @ | IN SOA | linuxprobe.com. | root.linuxprobe.com. | ( |
| | | | | 0;serial |
| | | | | 1D;refresh |
| | | | | 1H;retry |
| | | | | 1W;expire |
| | | | | 3H);minimum |
| | NS | ns.linuxprobe.com. | | |
| ns | A | 192.168.10.10 | | |
| 10 | PTR | ns.linuxprobe.com. | #PTR 为指针记录，仅用于反向解析中。 | |
| 10 | PTR | mail.linuxprobe.com. | | |
| 10 | PTR | www.linuxprobe.com. | | |
| 20 | PTR | bbs.linuxprobe.com. | | |



**第 3 步:检验解析结果。**

重启 named 服务程序，让配置文件立即生效：

[root@linuxprobe ~]# systemctl restart named

执行 nslookup 命令检查反向解析结果：

[root@linuxprobe ~]# nslookup
> 192.168.10.10
Server: 127.0.0.1
Address: 127.0.0.1#53
10.10.168.192.in-addr.arpa name = ns.linuxprobe.com.
10.10.168.192.in-addr.arpa name = www.linuxprobe.com.
10.10.168.192.in-addr.arpa name = mail.linuxprobe.com.

> 192.168.10.20
Server: 127.0.0.1
Address: 127.0.0.1#53
20.10.168.192.in-addr.arpa name = bbs.linuxprobe.com.

## 12.4 部署从服务器

真实网络环境中一台主服务器往往不能满足所有用户的需求，”从服务器”可以从主服务器上抓取指定的区域数据文件，起到备份解析记录与负载均衡的作用，配置过程大致流程：

1:在主服务器的区域信息文件中允许该从服务器的更新请求，并重新加载配置文件。

2:在从服务器中填写主服务器地址与要抓取的区域信息，并重新加载配置文件。

3:在从服务器中查看/var/named/slaves 目录或使用 nslookup 验证试验结果。

**试验环境中主机名称与 IP 地址（两台）**

| 主机名称 | 操作系统 | IP 地址 |
|---|---|---|
| 主服务器 | 红帽 RHEL7 操作系统 | 192.168.10.10 |
| 从服务器 | 红帽 RHEL7 操作系统 | 192.168.10.20 |

**第 1 步:修改主服务器中区域信息文件：**

allow-update { 允许更新区域信息的主机地址;};

[root@linuxprobe ~]# vim /etc/named.rfc1912.zones

```
zone "linuxprobe.com" IN {
type master;
file "linuxprobe.com.zone";
allow-update { 192.168.10.20; };
};

zone "10.168.192.in-addr.arpa" IN {
type master;
file "192.168.10.arpa";
allow-update { 192.168.10.20; };
```

重启 named 服务程序，让配置文件立即生效：

[root@linuxprobe ~]# systemctl restart named

**第 2 步:修改从服务器中的区域信息文件。**

必须将主配置文件"/etc/named.conf"中的监听地址与允许查询地址修改为 any，另外在区域文件(named.rfc1912.zones)中默认已存在一些注释内容与区域信息，可不必理会，只需将新的区域信息写到后面即可：

```
[root@linuxprobe ~]# vim /etc/named.rfc1912.zones
zone "linuxprobe.com" IN {
//请注意服务类型必需是 slave，而不能是 master。
type slave;
//指定主 DNS 服务器的 IP 地址。
masters { 192.168.10.10; };
//此为缓存到区域文件后保存的位置和名称。
file "slaves/linuxprobe.com.zone";
};
zone "10.168.192.in-addr.arpa" IN {
type slave;
masters { 192.168.10.10; };
file "slaves/192.168.10.arpa";
};
```

因为在红帽 RHCSA、RHCE 或 RHCA 考试后都要重启您的实验机再执行判分脚本。

所以请读者在日常工作中也要记得将需要的服务加入到开机启动项中:"systemctl enable named"。

**第 3 步:验证试验成果（请确认 DNS 地址为 192.168.10.20)。**

重启 named 服务程序，让配置文件立即生效：

```
[root@localhost named]# systemctl restart named
```

当前已经在 named 服务的数据文件目录了：

```
[root@localhost named]# pwd
/var/named
```

果然在 slaves 目录中出现了主服务器中的区域文件：

```
[root@localhost named]# ls slaves/
192.168.10.arpa linuxprobe.com.zone
```

使用 nslookup 命令看看解析能否成功吧：

```
[root@localhost named]# nslookup
> www.linuxprobe.com
Server: 192.168.10.20
Address: 192.168.10.20#53
Name: www.linuxprobe.com
Address: 192.168.10.10

> 192.168.10.10
Server: 192.168.10.20
Address: 192.168.10.20#53
10.10.168.192.in-addr.arpa name = www.linuxprobe.com.
10.10.168.192.in-addr.arpa name = ns.linuxprobe.com.
10.10.168.192.in-addr.arpa name = mail.linuxprobe.com.
```

## 12.5 安全的加密传输

DNS 服务是互联网的基础建设设施，几乎所有的网络应用都依赖于 DNS 服务做出的查询结果，如果互联网中的 DNS 服务不能正常提供解析服务，那么即使 Web 或 Email 服务都运行正常，也无法让用户顺利使用到它们了。

13 台根 DNS 服务器以及互联网中的 DNS 服务器绝大多数(超过 95%)是基于 BIND 服务程序搭建的，BIND 服务程序为了能够安全的提供解析服务而支持了 TSIG(**TSIG**RFC 2845)加密机制，TSIG 主要是利用密码编码方式保护区域信息的传送(Zone Transfer)，**也就是说保证了 DNS 服务器之间传送区域信息的安全。**

TSIG 仅有一组密码，而不区分公/私钥，所以一般只会分配给可信任的从服务器。

本实验基于上面的主服务器与从服务器的配置，请读者自行准备 DNS 实验环境，IP 地址要求如下：

| 主机名称 | 操作系统 | IP 地址 |
|---|---|---|
| 主服务器 | 红帽 RHEL7 操作系统 | 192.168.10.10 |
| 从服务器 | 红帽 RHEL7 操作系统 | 192.168.10.20 |

书接上章，重新启动 named 服务后可以看到 slaves 目录中出现区域数据文件。

```
[root@linuxprobe ~]# systemctl restart named
[root@linuxprobe ~]# ls -al /var/named/slaves/
total 12
drwxrwx---. 2 named named 54 Jun 7 16:02 .
drwxr-x---. 6 root named 4096 Jun 7 15:58 ..
-rw-r--r--. 1 named named 432 Jun 7 16:02 192.168.10.arpa
-rw-r--r--. 1 named named 439 Jun 7 16:02 linuxprobe.com.zone
```

第 1 步：在主服务器中生成密钥

**dnssec-keygen 命令用于生成安全的 DNS 服务密钥，格式为：”dnssec-keygen [参数] “。**

| 参数 | 作用 |
|---|---|
| -a | 指定加密算法（包括:RSAMD5 (RSA)、RSASHA1、DSA、NSEC3RSASHA1、NSEC3DSA 等） |
| -b | 密钥长度(HMAC-MD5 长度在 1-512 位之间) |
| -n | 密钥的类型(HOST 为与主机相关的) |

密钥参数：128 位 HMAC-MD5 算法，主机名称叫做 master-slave。

```
[root@linuxprobe ~]# dnssec-keygen -a HMAC-MD5 -b 128 -n HOST master-slave
Kmaster-slave.+157+46845
```

查看下生成出的密钥文件(依次为公钥与密钥)：

```
[root@linuxprobe ~]# ls -al Kmaster-slave.+157+46845.*
-rw-------. 1 root root 56 Jun 7 16:06 Kmaster-slave.+157+46845.key
-rw-------. 1 root root 165 Jun 7 16:06 Kmaster-slave.+157+46845.private
```

查看私钥内容（**把 Key 的值记录下来**）：

```
[root@linuxprobe ~]# cat Kmaster-slave.+157+46845.private
Private-key-format: v1.3
Algorithm: 157 (HMAC_MD5)
Key: 1XEEL3tG5DNLOw+1WHfE3Q==
Bits: AAA=
```

Created: 20150607080621

Publish: 20150607080621

Activate: 20150607080621

**第 2 步：在主服务器上创建密钥验证文件：**

[root@linuxprobe ~]# vim /var/named/chroot/etc/transfer.key

//依次为密钥名称、密钥加密类型以及私钥的 Key 值。

key "master-slave" {

algorithm hmac-md5;

secret "1XEEL3tG5DNLOw+1WHfE3Q==";

};

设置 transfer.key 文件的所有者和组：

[root@linuxprobe ~]# chown root.named /var/named/chroot/etc/transfer.key

为了更加的安全，设置权限为 640（rw-r——-）：

[root@linuxprobe ~]# chmod 640 /var/named/chroot/etc/transfer.key

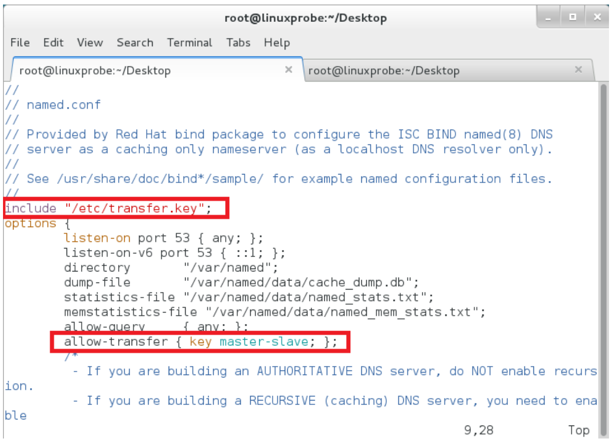
将密钥文件做硬链接到/etc 目录中：

[root@linuxprobe ~]# ln /var/named/chroot/etc/transfer.key /etc/transfer.key

**第 3 步：开启主服务器的密钥验证功能。**

开启密钥验证功能（修改如下图）：

[root@linuxprobe ~]# vim /etc/named.conf




**第 4 步：验证试验成果（无法获得区域数据信息了）。**

将从服务器之前获取到的区域数据文件都删除掉：

[root@linuxprobe ~]# rm -rf /var/named/slaves/*

重新启动 named 服务程序：

[root@linuxprobe ~]# systemctl restart named

果然此时已经无法获取到区域数据文件了：

[root@linuxprobe ~]# ls -al /var/named/slaves/

total 4

drwxrwx---. 2 named named 6 Jun 7 17:17 .

drwxr-x---. 6 root named 4096 Jun 7 15:58 ..

**第 5 步:配置从服务器支持密钥验证。**

先将密钥文件从主服务器中传送到从服务器：

[root@linuxprobe ~]# scp /var/named/chroot/etc/transfer.key root@192.168.10.20:/var/named/chroot/etc

The authenticity of host '192.168.10.20 (192.168.10.20)' can''t be established.

ECDSA key fingerprint is 4f:a7:91:9e:8d:6f:b9:48:02:32:61:95:48:ed:1e:3f.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '192.168.10.20' (ECDSA) to the list of known hosts.

root@192.168.10.20's password:此处输入对方主机的 root 用户密码。

transfer.key 100% 81 0.1KB/s 00:00

进入到从服务器的 named 服务的数据目录中：

[root@linuxprobe ~]# cd /var/named/chroot/etc

查看下刚刚 transfer.key 有没有成功传送过来：

[root@localhost etc]# ls -al transfer.key

-rw-r-----. 1 root root 81 Jun 7 17:20 transfer.key

修改文件的所有者和所有组：

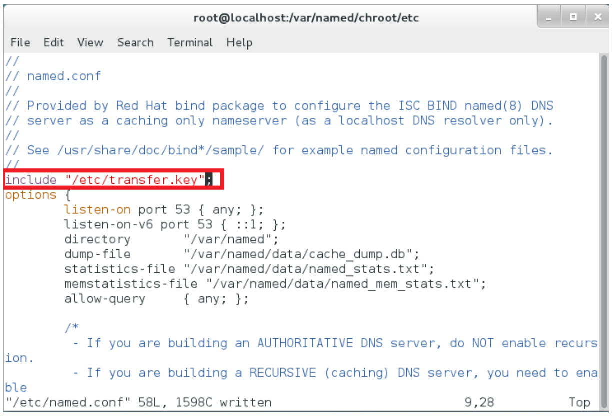
[root@localhost etc]# chown root:named transfer.key

创建文件链接：

[root@localhost etc]# ln transfer.key /etc/transfer.key

编辑主配置文件设置支持密钥验证（有两处配置，请看下图）。

[root@localhost etc]# vim /etc/named.conf

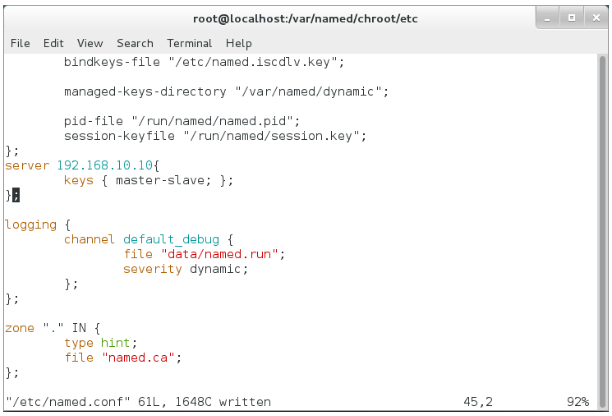



请先不要着急退出!!!请在大约 43 行部分追加以下内容：

server 主服务器 IP 地址 {

keys {密钥名称;};

};




因为在红帽 RHCSA、RHCE 或 RHCA 考试后都要重启您的实验机再执行判分脚本。

所以请读者在日常工作中也要记得将需要的服务加入到开机启动项中:" systemctl enable named "。

**第 6 步:验证试验成果(成功获取区域数据信息)。**

重新启动从服务器的 named 服务程序：

[root@linuxprobe ~]# systemctl restart named
果然又在 slaves 目录中看到了区域数据文件了:
[root@linuxprobe ~]# ls -al /var/named/slaves/
total 12
drwxrwx---. 2 named named 54 Jun 7 17:29 .
drwxr-x---. 6 root named 4096 Jun 7 15:58 ..
-rw-r--r--. 1 named named 432 Jun 7 17:29 192.168.10.arpa
-rw-r--r--. 1 named named 439 Jun 7 17:29 linuxprobe.com.zone

## 12.6 部署缓存服务器

**缓存服务器**(Caching DNS Server)是一种不负责域名数据维护，也不负责域名解析的 DNS 服务类型，缓存服务器是将用户经常使用到得域名与 IP 地址解析记录保存在主机本地中，提升下次解析的效率，所以一般用于对高品质上网有需求的内网之中，配置流程为:

第 1 步:配置系统的双网卡参数。
第 2 步:在主配置文件中添加缓存转发参数。
第 3 步:重启 DNS 服务后验证成果。

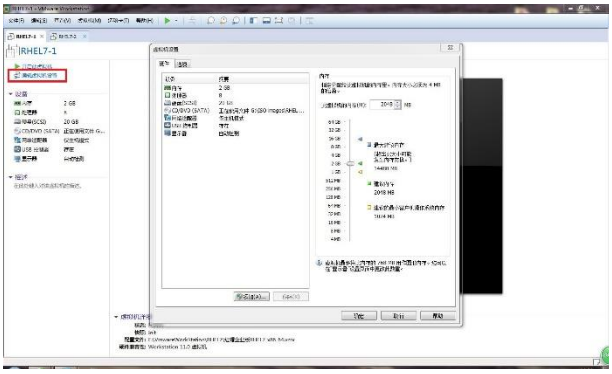
**试验环境中主机名称与 IP 地址（两台）**

| 主机名称 | 操作系统 | IP 地址 |
| --- | --- | --- |
| 缓存服务器 | 红帽 RHEL7 操作系统 | 网卡(外网):根据实际情况 DHCP 或手工指定 IP 地址与网关等信息。 |
| | | 网卡(内网):192.168.10.10 |
| 客户端 | 红帽 RHEL7 操作系统 | 192.168.10.20 |

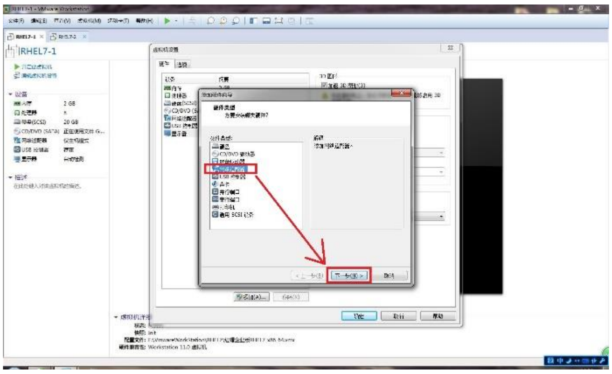
**第 1 步:配置系统的双网卡参数。**

如前面介绍的缓存服务器一般用于企业内网中，起到减少内网用户查询 DNS 的消耗，那么为了更加的贴近实际网络环境、实现外网查询功能，需要为缓存服务器中再添加一块网卡。

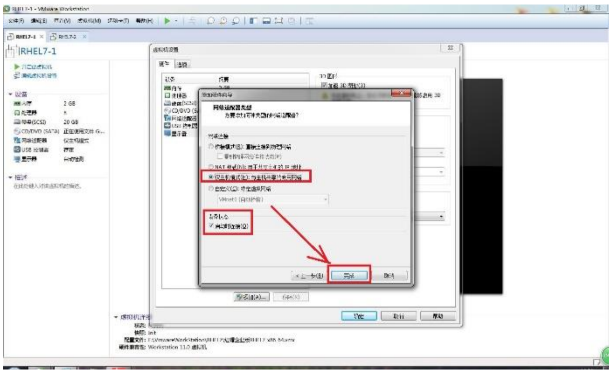
第 1 步：点击“编辑虚拟机设置”。



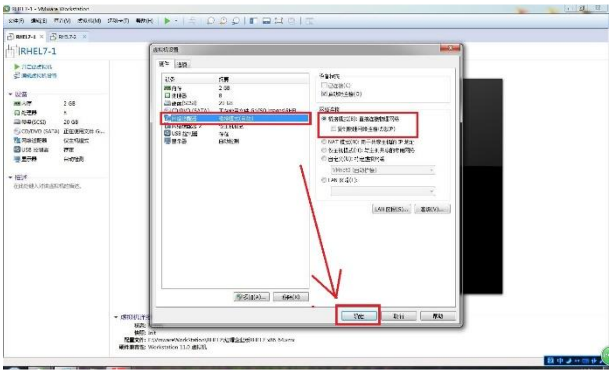
第 2 步：添加新的设备——“网络适配器”



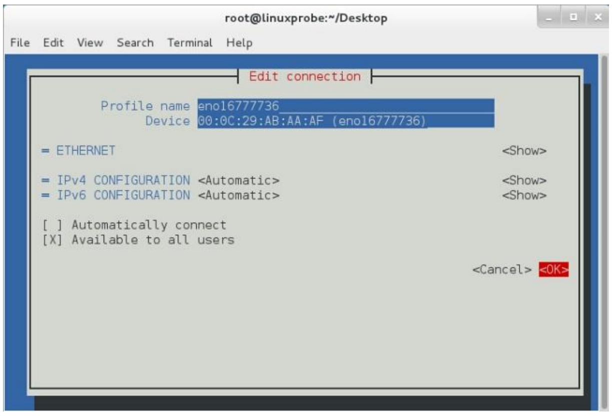
第 3 步：将网络类型修改为“仅主机模式”



第 4 步：将原先第一块网卡类型修改为“桥接模式”

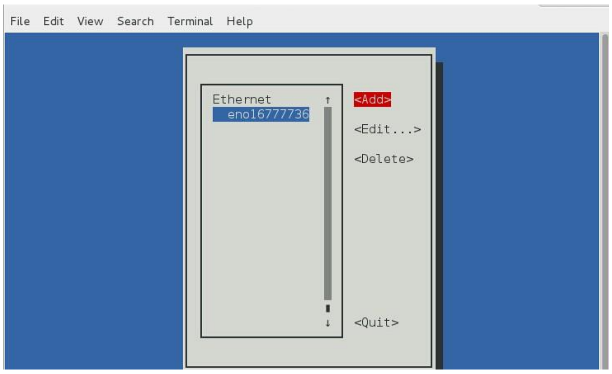


使用 nmtui 工具将第 1 块网卡的 IP 地址方法修改为 dhcp 模式（**请读者根据实际上网环境来设置**）：

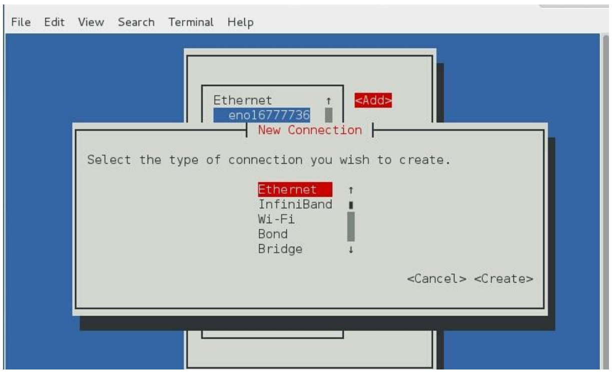



新添加的网卡(第 2 块)在 nmtui 工具列表中还没有显示，需要聪明的读者们动手添加下：

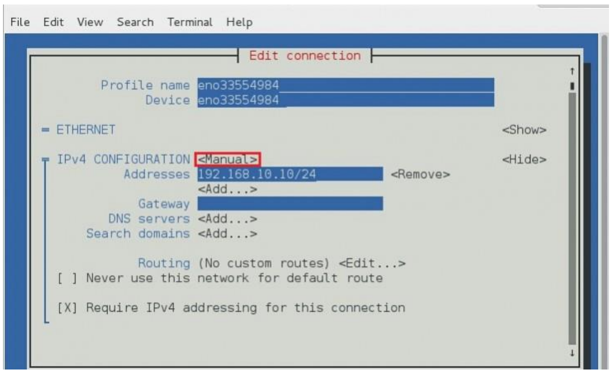
第 1 步：添加一块新的网卡设备。



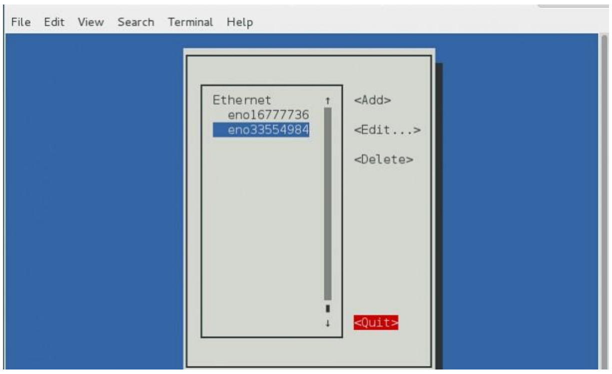

第 2 步：选择网络链接类型。



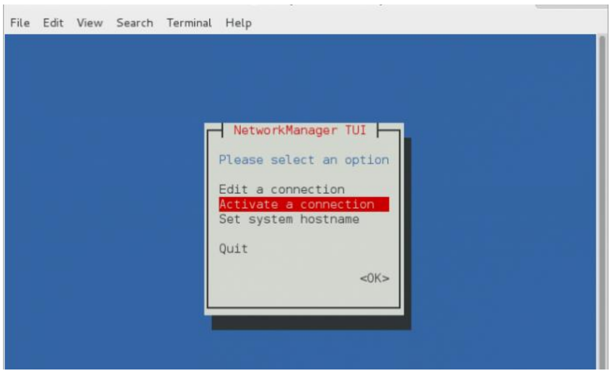

第 3 步：填写网络名称与 IP 地址等信息



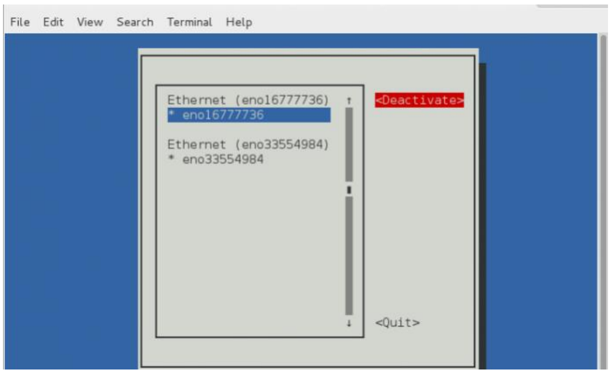

第 4 步：看到添加成功后退出即可。



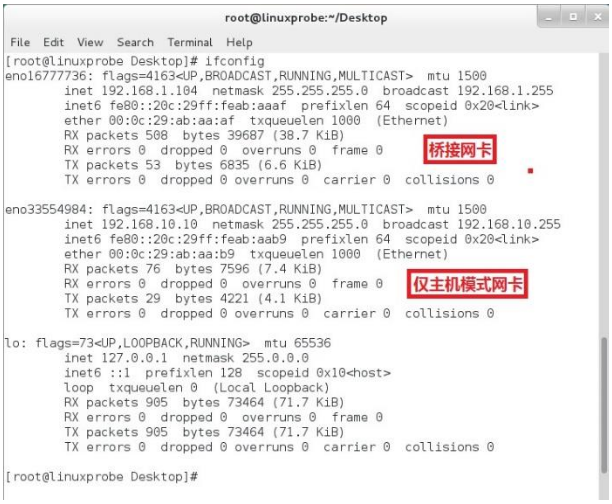
第 5 步：点击“激活一个网络链接”。




第 6 步：激活后退出，再次查看网卡信息。




**当各位读者配置成功后网卡应该都会显示出正确的 IP 地址啦：**




**第 2 步:在主配置文件中添加缓存转发参数。**

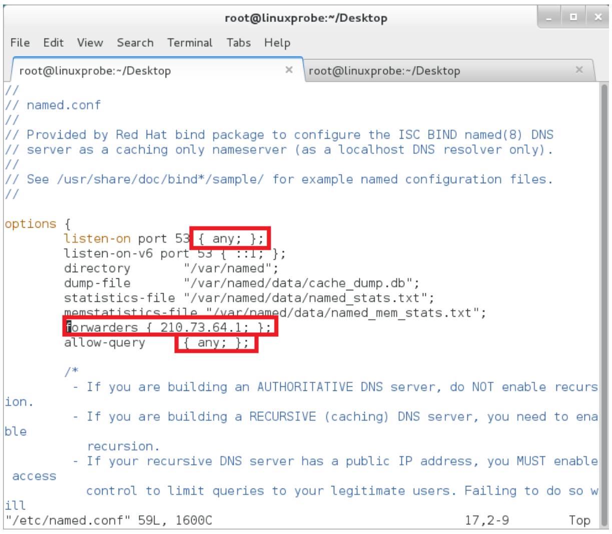
缓存服务器的配置步骤非常简单，首先安装 bind 服务(yum install named -y)，然后编辑主配置文件：

[root@linuxprobe ~]# vim /etc/named.conf

将监听 IP 端口与允许查询主机修改为 any,再添加一行” forwarders { 上游 DNS 服务器地址; };”

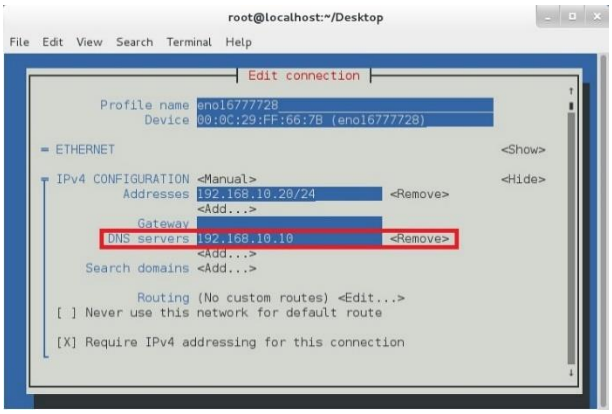
上游 DNS 服务器地址指的是从何处取得区域数据文件，主要对比查询速度、稳定性、安全性等因素。

本次使用北京市 DNS 服务器:”210.73.64.1″，请读者选择前先 Ping 下能否通信，否则可能会导致解析失败!!




**第 3 步:重启 DNS 服务后验证成果。**

将客户端的网卡 DNS 地址指向**缓存服务器**(192.168.10.10)，配置网卡应该已经很熟练，很简单了吧:




修改完主配置文件后请将 named 服务重启:

[root@linuxprobe ~]# systemctl restart named

使用 nslookup 命令验证实验成果(如果解析不成功，请读者多留意下是不是上游 DNS 选择的问题):

[root@linuxprobe ~]# nslookup

> www.linuxprobe.com

Server: 192.168.10.10

Address: 192.168.10.10#53

Non-authoritative answer:

Name: www.linuxprobe.com

Address: 113.207.76.73

Name: www.linuxprobe.com

Address: 116.211.121.154

再来尝试下反向查询吧（此为 google 的免费 DNS 服务器地址）:

> 8.8.8.8

Server: 192.168.10.10

Address: 192.168.10.10#53

Non-authoritative answer:

8.8.8.8.in-addr.arpa name = google-public-dns-a.google.com.
Authoritative answers can be found from:
in-addr.arpa nameserver = f.in-addr-servers.arpa.
in-addr.arpa nameserver = b.in-addr-servers.arpa.
in-addr.arpa nameserver = a.in-addr-servers.arpa.
in-addr.arpa nameserver = e.in-addr-servers.arpa.
in-addr.arpa nameserver = d.in-addr-servers.arpa.
in-addr.arpa nameserver = c.in-addr-servers.arpa.
a.in-addr-servers.arpa internet address = 199.212.0.73
a.in-addr-servers.arpa has AAAA address 2001:500:13::73
b.in-addr-servers.arpa internet address = 199.253.183.183
b.in-addr-servers.arpa has AAAA address 2001:500:87::87
c.in-addr-servers.arpa internet address = 196.216.169.10
c.in-addr-servers.arpa has AAAA address 2001:43f8:110::10
d.in-addr-servers.arpa internet address = 200.10.60.53
d.in-addr-servers.arpa has AAAA address 2001:13c7:7010::53
e.in-addr-servers.arpa internet address = 203.119.86.101
e.in-addr-servers.arpa has AAAA address 2001:dd8:6::101
f.in-addr-servers.arpa internet address = 193.0.9.1
f.in-addr-servers.arpa has AAAA address 2001:67c:e0::1

### 12.7 分离解析技术

假如喜欢《Linux 就该这么学》的海外留学生越来越多，但网站服务器架设在北京市，那么留学生访问起来速度一定很慢，而将服务器架设在美国，那又会让国内用户访问变得很麻烦，于是我们便可以采用分离解析的办法，虽然访问的是相同的网址，但国内用户访问北京服务器，而留学生则直接访问美国服务器。

**分离解析**:当来自于不同 IP 地址的用户查询相同域名时会为其提供不同的解析结果，大致流程为：

第 1 步：在区域信息文件中填写不同的 Zone 区域信息。
第 2 步：建立独立的区域数据文件。
第 3 步：重新启动 named 服务并验证结果。

那么为了解决《Linux 就该这么学》访问速度的问题，站务管理员已经在美国架设好了网站服务器，请您部署 DNS 服务器并实现分离解析功能，北京用户与美国用户访问相同域名时解析出不同的 IP 地址，拓扑如下：DNS 分离解析拓扑




| 主机名称 | 操作系统 | IP 地址 |
| --- | --- | --- |

| | | |
|---|---|---|
| DNS 服务器 | 红帽 RHEL7 操作系统 | 北京网络:122.71.115.10 |
| | | 美国网络:106.185.25.10 |
| 北京用户 | Windows7 | 122.71.155.1 |
| 海外用户 | Windows7 | 106.185.25.1 |

请读者先动手安装下 BIND 服务(“**yum install bind-chroot -y**“)，并将其加入到开机启动项中。

**第 1 步：在区域信息文件中填写不同的 Zone 区域信息。**

修改主配置文件”**/etc/named.conf**“，将监听端口与允许查询主机修改为 **any，并将约在 51 行的根域信息删除掉**:

zone "." IN {
type hint;
file "name.ca";
};

编辑区域信息文件”**/etc/named.rfc1912.zones**“，**清空该文件所有默认的数据**并添加以下内容：

//ACL 定义了 china 与 american 分别对应的 IP 地址，以下就不需要写 IP 地址了。
acl "china" { 122.71.115.0/24; };
acl "american" { 106.185.25.0/24;};
//匹配所有 china 内的 IP 地址，对应的域名数据文件为 linuxprobe.com.china。
view "china"{
match-clients { "china"; };
zone "linuxprobe.com" {
type master;
file "linuxprobe.com.china";
};
};
//匹配所有 american 内的 IP 地址，对应的域名数据文件为 linuxprobe.com.american。
view "american" {
match-clients { "american"; };
zone "linuxprobe.com" {
type master;
file "linuxprobe.com.american";
};
};

这样来自不同 IP 地址的用户访问 linuxprobe.com 域时就会访问不同的区域数据文件，从而达到了分离解析的作用。

**第 2 步：建立独立的区域数据文件。**

切换工作目录到 named 服务目录中：

[root@linuxprobe ~]# cd /var/named

分别复制出两份域名区域文件数据的模板：

[root@linuxprobe named]# cp -a named.localhost linuxprobe.com.china
[root@linuxprobe named]# cp -a named.localhost linuxprobe.com.american

编辑对中国用户有效的域名区域数据文件 **vim linuxprobe.com.china**：

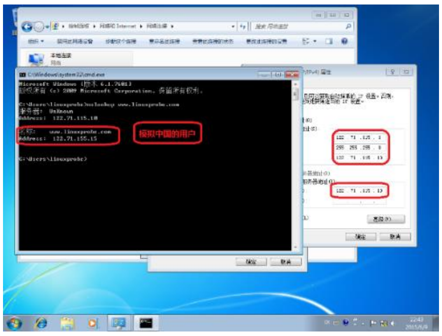
| $TTL 1D | #生存周期为 1 天 | | | | |
|---|---|---|---|---|---|
| @ | IN SOA | linuxprobe.com. | root.linuxprobe.com. | ( | |
| | #授权信息开始: | #DNS 区域的地址 | #域名管理员的邮箱(不要用@符号) | | |
| | | | | 0;serial | #更新序列号 |
| | | | | 1D;refresh | #更新时间 |
| | | | | 1H;retry | #重试延时 |
| | | | | 1W;expire | #失效时间 |
| | | | | 3H;minimum | #无效解析记录的缓存时间 |
| | NS | ns.linuxprobe.com. | | #域名服务器记录 | |
| ns | IN A | 122.71.155.10 | | #地址记录(ns.linuxprobe.com.) | |
| www | IN A | 122.71.155.15 | | #地址记录(www.linuxprobe.com.) | |

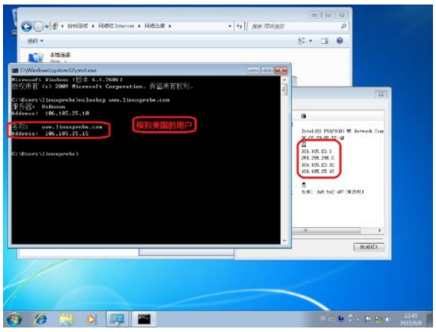
编辑对美国用户有效的域名区域数据文件 **vim linuxprobe.com.american**：

| $TTL 1D | #生存周期为 1 天 | | | | |
|---|---|---|---|---|---|
| @ | IN SOA | linuxprobe.com. | root.linuxprobe.com. | ( | |
| | #授权信息开始: | #DNS 区域的地址 | #域名管理员的邮箱(不要用@符号) | | |
| | | | | 0;serial | #更新序列号 |
| | | | | 1D;refresh | #更新时间 |
| | | | | 1H;retry | #重试延时 |
| | | | | 1W;expire | #失效时间 |
| | | | | 3H;minimum | #无效解析记录的缓存时间 |
| | NS | ns.linuxprobe.com. | | #域名服务器记录 | |
| ns | IN A | 106.185.25.10 | | #地址记录(ns.linuxprobe.com.) | |
| www | IN A | 106.185.25.15 | | #地址记录(www.linuxprobe.com.) | |

**第 3 步：重新启动 named 服务并验证结果。**

开启两台 windows7 系统的虚拟机，分别模拟中国与美国 IP 地址后执行”**nslookup** www.linuxprobe.com“，



第 13 章 使用 DHCP 动态管理主机地址。

**章节简述：**

DHCP 协议服务能够自动化的管理局域网内的主机 IP 地址，有效的提升 IP 地址使用率，提高配置效率，减少管理与维护成本。学习 dhcpd 服务程序的使用方法并逐条讲解配置参数，完整演示自动化分配 IP 地址、绑定 IP 地址与 mac 地址等实验。DHCP 中继代理技术是多个物理网段共同一台 DHCP 服务器的最佳解决方案，运维人员必学的实用技术之一。

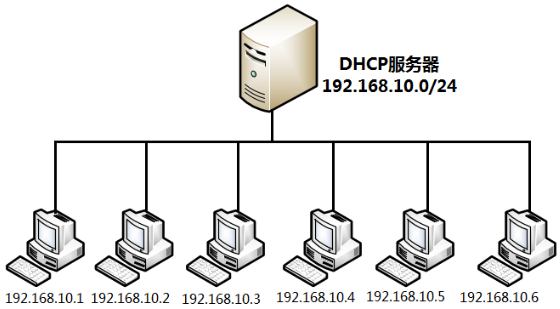
## 13.1 动态主机管理协议

**DHCP 动态主机管理协议**(Dynamic Host Configuration Protocol)是一种基于 UDP 协议且仅限用于局域网的网络协议，主要用途是**为局域网内部设备或网络供应商自动分配 IP 地址**，通常会应用在大型的局域网环境中或局域网内存在比较多的移动办公设备，DHCP 协议能够实现集中的管理、分配 IP 地址。

DHCP 服务程序能够使局域网内的主机自动且动态的获取 IP 地址、子网掩码、网关地址以及 DNS 服务器地址等信息，且能够有效的提升地址使用率，提高配置效率，减少管理和维护成本。




DHCP 协议能够保证任何 IP 地址在同一时刻只能由一台 DHCP 客户机使用，且能够为指定主机分配固定的 IP 地址。




**DHCP 服务程序的常见术语：**

**作用域**:一个完整的 IP 地址段，DHCP 服务根据作用域来管理网络的分布、分配 IP 地址及其他配置参数。

**超级作用域**:用于支持同一物理网络上多个逻辑 IP 地址子网段，包含作用域的列表，并对子作用域统一管理。

**排除范围**:将某些 IP 地址在作用域中排除，确保这些 IP 地址不会被提供给 DHCP 客户机。

**地址池**:在定义 DHCP 服务的作用域并应用排除范围后，剩余用来动态分配给 DHCP 客户机的 IP 地址范围。

**租约**:即 DHCP 客户机能够使用动态分配到的 IP 地址的时间。

**预约**:保证局域子网中特定设备总是获取到相同的 IP 地址。

## 13.2 安装 dhcpd 服务程序

dhcpd 服务程序用于提供 DHCP 协议服务，确认镜像挂载且 yum 仓库配置完毕后即可开始安装：

```
[root@linuxprobe ~]# yum install dhcp
> Package dhcp.x86_64 12:4.2.5-27.el7 will be installed
………………省略部分安装过程………………
Complete!
```

**dhcpd 服务程序与配置文件：**

| 主配置文件 | /etc/dhcp/dhcpd.conf |
| --- | --- |
| 执行程序 | /usr/sbin/dhcpd<br>/usr/sbin/dhcrelay |

先来分析下 dhcp 程序的主配置文件吧:

[root@linuxprobe ~]# cat /etc/dhcp/dhcpd.conf

# DHCP Server Configuration file.

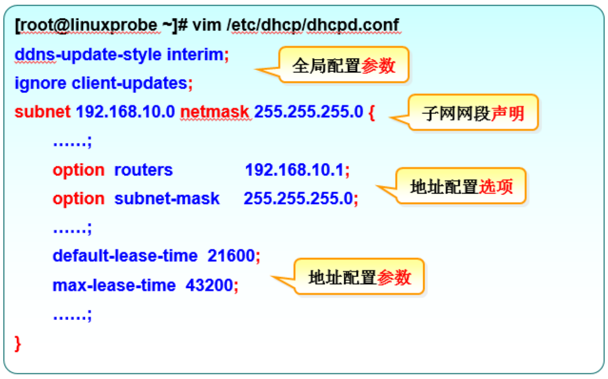
# see /usr/share/doc/dhcp*/dhcpd.conf.example

# see dhcpd.conf(5) man page

是的，你没有看错！dhcpd 服务程序的配置文件默认只有注释语句，需要参考下模板文件:

[root@linuxprobe ~]# cat /usr/share/doc/dhcp*/dhcpd.conf.example

一个标准的 DHCP 配置文件应该包括**全局配置参数、子网网段声明、地址配置选项**以及**地址配置参数**:




全局配置参数用于定义整个配置文件的全局参数，而子网网段声明用于配置整个子网段的地址属性，具体参数有:

| 参数 | 作用 |
| --- | --- |
| ddns-update-style 类型 | 定义 DNS 服务动态更新的类型，类型包括:<br>none（不支持动态更新），interim（互动更新模式）与 ad-hoc(特殊更新模式)。 |
| allow/ignore client-updates | 允许/忽略客户机更新 DNS 记录。 |
| default-lease-time 21600 | 默认超时时间。 |
| max-lease-time 43200 | 最大超时时间。 |
| option domain-name-servers 8.8.8.8 | 定义 DNS 服务器地址。 |
| option domain-name "domain.org" | 定义 DNS 域名。 |
| range | 定义用于分配的 IP 地址池。 |
| option subnet-mask | 定义客户机的子网掩码。 |
| option routers | 定义客户机的网关地址。 |
| broadcase-address 广播地址 | 定义客户机的广播地址。 |

| | |
|---|---|
| ntp-server IP 地址 | 定义客户机的网络时间服务器（NTP)。 |
| nis-servers IP 地址 | 定义客户机的 NIS 域服务器的地址。 |
| hardware 硬件类型 MAC 地址 | 指定网卡接口的类型与 MAC 地址。 |
| server-name 主机名 | 通知 DHCP 客户机服务器的主机名。 |
| fixed-address IP 地址 | 将某个固定 IP 地址分配给指定主机。 |
| time-offset 偏移差 | 指定客户机与格林尼治时间的偏移差。 |

### 13.3 自动管理 IP 地址

DHCP 协议的初衷是更高效的集中管理管理局域网内 IP 地址资源，那么符合让 DHCP 服务更准确的完成工作呢?

**为了让实验更有挑战性，我们来模拟一个真实环境吧:**

运维部:

明日约有 50 名外部学员自带笔记本设备来我司培训学习，请保证学员能够用 DHCP 获取 IP 地址并正常上网。

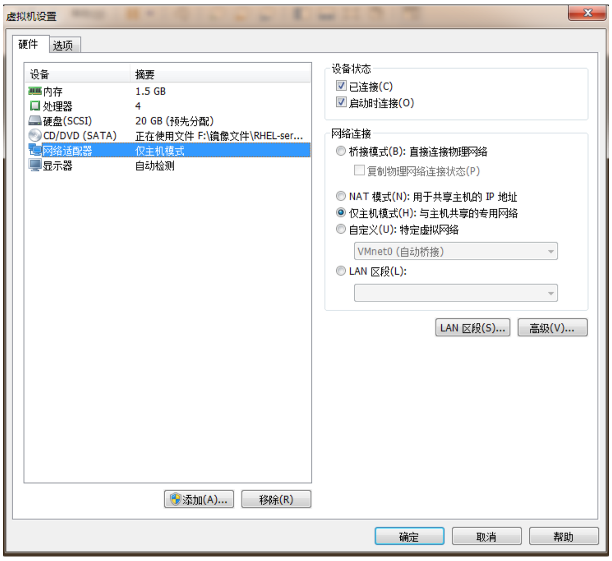
**机房网段及参数如下:**

| 参数名称 | 值 |
|---|---|
| 默认租约时间 | 21600 秒 |
| 最大租约时间 | 43200 秒 |
| IP 地址范围 | 192.168.10.50~192.168.10.150 |
| 子网掩码 | 255.255.255.0 |
| 网关地址 | 192.168.10.1 |
| DNS 服务地址 | 192.168.10.1 |
| 搜索域 | linuxprobe.com |

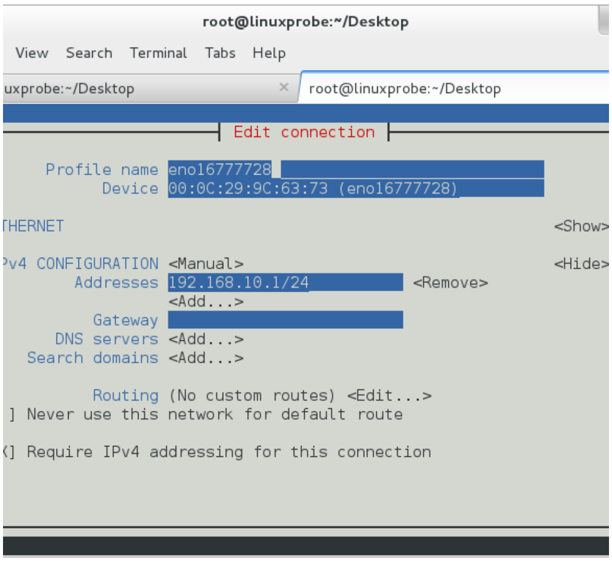
**看完配置要求后，首先请准备实验环境:**

| 主机类型 | 操作系统 | IP 地址 |
|---|---|---|
| DHCP 服务端 | 红帽 RHEL7 操作系统 | 192.168.10.1 |
| DHCP 客户机 | 红帽 RHEL7 操作系统 | DHCP 自动获取地址 |

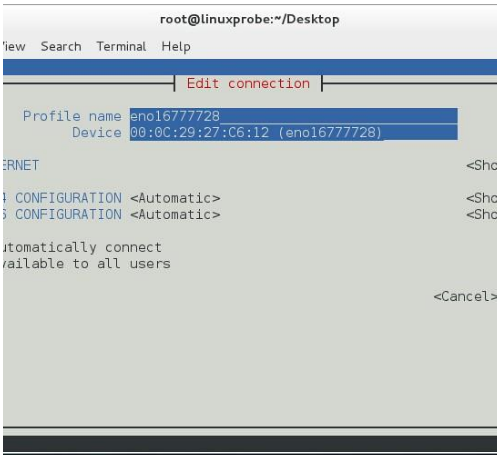
配置虚拟机网络类型（两台主机都要配置成一样的）




配置 DHCP 服务端的 IP 地址。

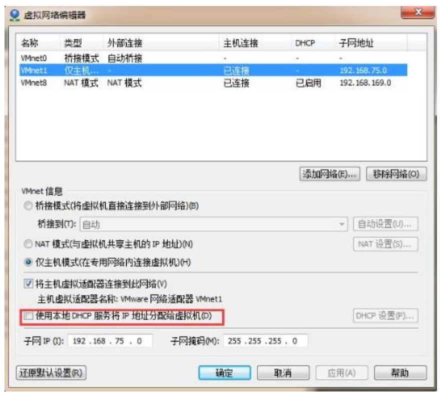



配置 DHCP 客户机的网卡。




**另外因为虚拟机软件 VmwareWorkstation 默认开启了虚拟机 DHCP 服务，必需关闭后再进行 DHCP 实验：**






**当您确定两台主机的网卡和虚拟机都已经配置妥当，那么就开始配置 DHCP 服务程序吧：**

[root@linuxprobe ~]# vim /etc/dhcp/dhcpd.conf

//请注意 dhcpd 服务程序的配置文件中每个参数均需要以;号结尾。

//请将下面表格中的参数逐行写入到 dhcpd.conf 文件中。

| 参数 | 作用 |
| --- | --- |
| ddns-update-style none; | 设置 DHCP 服务不自动动态更新。 |

| | |
|---|---|
| ignore client-updates; | 忽略客户机更新 DNS 记录。 |
| subnet 192.168.10.0 netmask 255.255.255.0 { | 作用域为 192.168.10.0/24 网段。 |
| range 192.168.10.50 192.168.10.150; | IP 地址池为 192.168.10.50-150（约 100 个 IP 地址）。 |
| option subnet-mask 255.255.255.0; | 定义客户机默认的子网掩码。 |
| option routers 192.168.10.1; | 定义客户机的网关地址。 |
| option domain-name "linuxprobe.com"; | 定义默认的搜索域。 |
| option domain-name-servers 192.168.10.1; | 定义客户机的 DNS 地址。 |
| default-lease-time 21600; | 定义默认租约时间。 |
| max-lease-time 43200; | 定义最大预约时间。 |
| } | 此为结束符 |

重启 dhcpd 服务程序：

[root@linuxprobe ~]# systemctl start dhcpd

添加到开机启动项中：

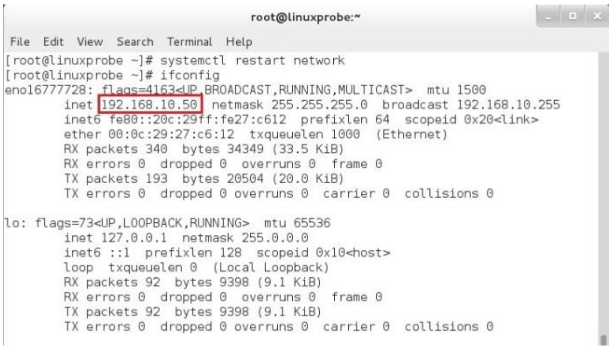
[root@linuxprobe ~]# *systemctl enable dhcpd*

ln -s '/usr/lib/systemd/system/dhcpd.service' '/etc/systemd/system/multi-user.target.wants/dhcpd.service'

因为在红帽 RHCSA、RHCE 或 RHCA 考试后都要重启您的实验机再执行判分脚本。

所以请读者在日常工作中也要记得将需要的服务加入到开机启动项中:”**systemctl enable dhcpd**“。

此时在 DHCP 客户机上重启网卡设备，即可自动获取到 IP 地址：



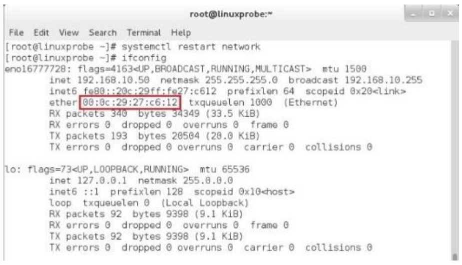

### 13.4 分配固定 IP 地址

DHCP 协议的术语**预约**指的就是保证局域子网中特定设备总是获取到相同的 IP 地址，换句话说 dhcpd 服务会将某个 IP 地址私藏下来，只有匹配到特定主机了才会拿出来分配，而要做 IP 地址与主机的绑定，需要使用下面的参数格式：

| host 主机名称 { | | | |
|---|---|---|---|
| | hardware | ethernet | 该主机的 MAC 地址; |
| | fixed-address | 欲指定的 IP 地址; | |

}

**查看到要绑定 IP 地址的主机 mac 地址（00:0c:29:27:c6:12）：**



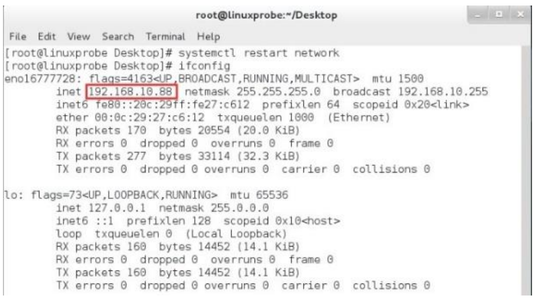

**在 DHCP 配置文件中添加绑定语句：**

ddns-update-style none;
ignore client-updates;
subnet 192.168.10.0 netmask 255.255.255.0 {
range 192.168.10.50 192.168.10.150;
option subnet-mask 255.255.255.0;
option routers 192.168.10.1;
option domain-name "linuxprobe.com";
option domain-name-servers 192.168.10.1;
default-lease-time 21600;
max-lease-time 43200;
host linuxprobe {
hardware ethernet 00:0c:29:27:c6:12;
fixed-address 192.168.10.88;
}
}

确定配置参数填写正确后重启 dhcpd 服务：

[root@linuxprobe ~]# systemctl start dhcpd

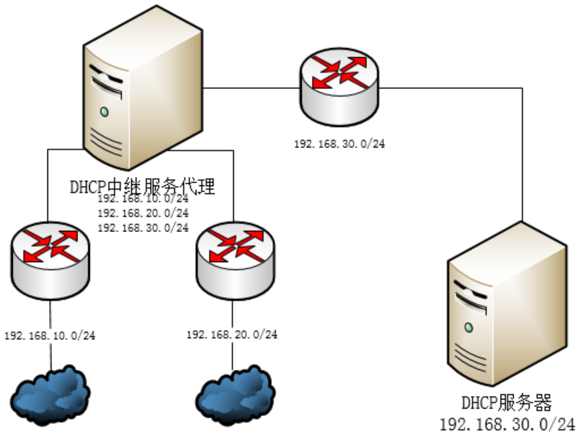
DHCP 客户机重新加载网卡设备后查看到顺利绑定到了指定的 IP 地址，不错哦~




## 13.5 DHCP 中继代理

**DHCP 中继代理**(即 **DHCP Relay** Agent)用于转发来自于另一个没有 DHCP 服务器子网段中客户端的 DHCP 请求，即当一台 DHCP 客户端发起请求后，此时 DHCP 中继代理就会将已经预先定义好的 DHCP 服务器的信息转发给客户

端。




如果客户机与 DHCP 服务器处在同一个子网段，则客户机自然能够顺利动态获取到 IP 地址，但若客户机与 DHCP 服务器处在不同的子网段或物理网段，则需要 DHCP Relay Agent 来处理和转发 DHCP 协议信息，换句话说，DHCP 中继代理可以让每个物理子网不再必需配有一台 DHCP 服务器，而是将请求转发给指定的 DHCP 服务器。

**在 BOOTP 模式中执行 dhcrelay 服务(DHCPv4)：**

**第 1 步:复制服务程序**

[root@linuxprobe ~]# cp /lib/systemd/system/dhcrelay.service /etc/systemd/system/

**第 2 步:编辑服务程序**

[root@linuxprobe ~]# vim /etc/systemd/system/dhcrelay.service

在 ExecStart 参数中添加指定 DHCP 服务器地址

ExecStart=/usr/sbin/dhcrelay -d --no-pid 192.168.10.1

如果希望仅某个网卡专门用于监听 DHCP 请求，则在 ExecStart 选项中追加-i 参数，默认全部网卡均监听。

ExecStart=/usr/sbin/dhcrelay -d --no-pid 192.168.10.1 -i eno16777736

**第 3 步:启动 dhcrelay 中继服务程序。**

[root@linuxprobe ~]# systemctl --system daemon-reload

[root@linuxprobe ~]# systemctl restart dhcrelay

[root@linuxprobe ~]# systemctl enable dhcrelay

坦白讲，一般 DHCP 中继功能在日常工作使用较少，或由路由器负责 DHCP 中继功能，极少用红帽 Linux 系统搭建。

**本章结束，您可以在此写下笔记：**

第 14 章 使用 Postfix 与 Dovecot 收发电子邮件。

**章节概述:**

本章节从电子邮局系统的组成角色开始讲起，了解 MUA、MTA 与 MDA 的作用，熟悉熟悉 SMTP、POP3 与 IMAP4 邮局协议。学习 postfix 与 dovecot 服务程序的使用方法并逐条讲解配置参数，完整演示了部署基础电子邮局系统以及设置用户别名邮箱的方法。

## 14.1 电子邮局系统

1971 年由美国国防部资助的的 ARPANET 科研项目遇到了严峻问题——参于科研项目的科学家在不同的地方工作，不能及时的分享各自的研究成果，迫切的需要一种能够借助于网络且建立在计算机之间的传输数据的方法。 当时麻省理工学院 Ray Tomlinson 博士也是 ARPANET 项目的科研成员，当年秋天他使用软件 SNDMSG 向自己另一台电脑发出了人类历史上第一封 Email 邮件。



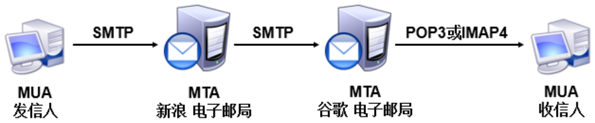
**Ray Tomlinson 博士决定选择”@”符号作为用户名与主机地址的间隔符。**

**邮件应用协议包括:**

简单邮件传输协议(SMTP)，用来发送或中转发出的电子邮件，占用 tcp 25 端口。

第三版邮局协议(POP3)，用于将服务器上把邮件存储到本地主机，占用 tcp 110 端口。

第四版互联网信息访问协议(IMAP4)，用于在本地主机上访问邮件，占用 tcp 143 端口。




**电子邮件系统**(E-mail，即 Electronic mail system)**由三部分组成:**

用户代理 MUA(Mail User Agent):用于收发邮件。

邮件传输代理 MTA(Mail Transfer Agent):将来自于 MUA 的邮件转发给指定用户。

邮件投递代理 MDA(Mail Delivery Agent):将来自于 MTA 的邮件保存到本机的收件箱中。

电子邮件系统与大多数的网络应用协议有本质的不同，例如前面讲过的文本传输协议(FTP)，FTP 服务程序就像拨打电话一样，需要对方当前也保持在线，否则会报错连接超时。但电子邮件的发送者则并不需要等待投递工作完成，因为如果对方服务器宕机了，则会将要发送的内容自动的暂时保存到本地，检测到对方服务器恢复后再次投递。另外如果您想搭建企业级的电子邮件系统，请考虑下面几点:

**反垃圾与反病毒模块**:阻止垃圾邮件或病毒邮件对企业邮箱的干扰。

**邮件加密**:保证邮件内容不被嗅探、篡改。

**邮件监控审核**:监控全体职员邮件中有无敏感词，透露企业资料等。

**稳定性**:有较好的防 DDOS 攻击的能力，保证系统在线率等。

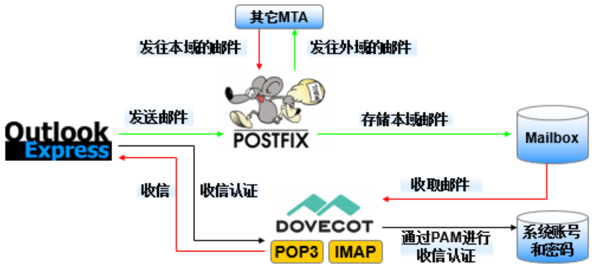
## 14.2 部署基础电子邮局系统

单独的使用 Postfix 服务程序并不能让用户完成收发邮件的操作，因为一个基础的电子邮局系统至少需要有 SMTP 服务器、POP3/IMAP 服务器，为了能够部署一个基础的电子邮局系统，我们需要使用到下面的软件：

**Postfix**:提供邮件发送服务，即 SMTP。

**Dovecot**:提供邮件收取服务，即 POP3。

**OutLook Express**:客户端收发邮件的工具。




Postfix(发送邮件)+Dovectot(接收邮件)+OutLook(客户端工具)

**配置本地主机名**

修改本地主机名的配置文件:

[root@linuxprobe ~] # vim /etc/hostname

mail.linuxprobe.com

[root@mail~] # hostname

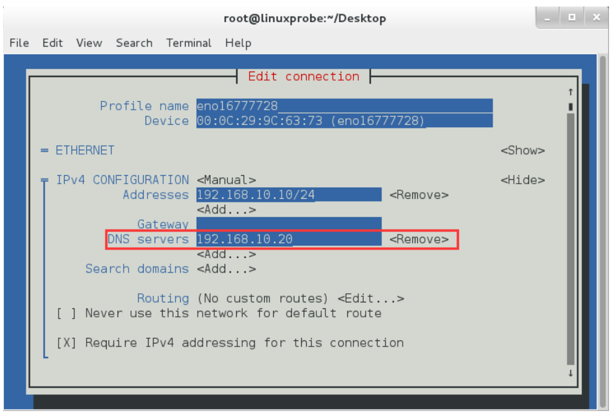
mail.linuxprobe.com

若要为用户提供 **linuxprobe 域**的电子邮局系统，则需先在 DNS 服务器中增加 A 记录和 MX 记录：

@ IN MX 10 mail.linuxprobe.com.

mail IN A 192.168.10.10

这样配置解析记录后，主机名即为 **mail.linuxprobe.com**，而邮件域为**@linuxprobe.com**。




**(请读者自行使用 nmtui 工具将网卡的 DNS 地址配置妥当即可)**

### 14.2.1 配置 Postfix 服务程序

Postfix 是一款由 IBM 出资研发的免费开源的邮局服务程序，兼容于 Sendmail 服务程序，即 Sendmail 用户可以很方便的迁移到 Postfix 程序，且收发件性能远超过 Sendmail，能够自动增加减少进程的数量，保证邮局系统的高性能与稳定性，另外 Postfix 是由诸多的小模块组成，每个小模块完成特定的功能，使得管理员可以灵活的组合这些模块。

**红帽 RHEL7 系统中默认已安装 postfix 邮局服务程序:**

[root@linuxprobe ~]#yum install postfix

Nothing to do

**Postfix 邮局服务程序的配置文件如下：**

| 文件 | 作用 |
| --- | --- |
| /usr/sbin/postfix | 主服务程序 |
| /etc/postfix/master.cf | master 主程序的配置文件。 |
| /etc/postfix/main.cf | postfix 服务的配置文件。 |
| /var/log/maillog | 记录邮件传递过程的日志。 |

**第 1 步:查看 Postfix 服务程序主配置文件:**

[root@mail~]# cat /etc/postfix/main.cf

配置文件足足有 679 行!但不用担心，绝大部分都是注释信息，我们只学习这些参数即可：

| 参数 | 作用 |
| --- | --- |
| myhostname | 邮局系统的主机名。 |
| mydomain | 邮局系统的域名。 |
| myorigin | 从本机寄出邮件的域名名称。 |
| inet_interfaces | 监听的网卡接口。 |
| mydestination | 可接收邮件的主机名或域名。 |
| mynetworks | 设置可转发那些主机的邮件。 |
| relay_domains | 设置可转发那些网域的邮件 |

编辑 Postfix 服务程序的主配置文件（修改 5 处参数，另外需要将参数前面的井号（#）去掉才可生效）：

[root@mail~] # vim /etc/postfix/main.cf

//修改第 76 行的邮局主机名。

myhostname = mail.linuxprobe.com

//修改第 83 行的邮局域名。

mydomain = linuxprobe.com

//修改第 99 行的寄出邮件域名，$mydomain 的值已在上面定义。

myorigin = $mydomain

//修改第 116 行的监听网卡。

inet_interfaces = all

//修改第 164 行的可接收邮件的主机名和域名。

mydestination = $myhostname, $mydomain

**第 2 步:创建邮局帐号：**

[root@mail~] # useradd boss

[root@mail~] # echo "linuxprobe" | passwd --stdin boss

Changing password for user boss. passwd: all authentication tokens updated successfully.

**第 3 步:启动 Postfix 服务程序:**

[root@mail~] # systemctl restart postfix

[root@mail~] # systemctl enable postfix

ln -s '/usr/lib/systemd/system/postfix.service' '/etc/systemd/system/multi-user.target.wants/postfix.service'

因为在红帽 RHCSA、RHCE 或 RHCA 考试后都要重启您的实验机再执行判分脚本。

所以请读者在日常工作中也要记得将需要的服务加入到开机启动项中:" **systemctl enable postfix** "。**14.2.2 配置 Dovecot 服务程序**

**第 1 步:安装 Dovecot 服务程序:**

[root@mail~] # yum install dovecot -y

Loaded plugins: langpacks, product-id, subscription-manager

………………省略部分安装过程………………

Installing: dovecot x86_64 1:2.2.10-4.el7 rhel7 3.2 M

Installing for dependencies: clucene-core x86_64 2.3.3.4-11.el7 rhel7 528 k

………………省略部分安装过程………………

Complete!

**第 2 步:修改 Dovecot 程序主配置文件:**

[root@mail~] # vim /etc/dovecot/dovecot.conf

//修改第 24 行的支持邮局协议。

protocols = imap pop3 lmtp

//然后追加允许明文认证(25 行)。

disable_plaintext_auth = no

//修改第 48 行的允许登陆网段地址,全部允许即为(0.0.0.0/0)。

login_trusted_networks = 192.168.10.0/24

**第 3 步:配置邮件的格式与存储路径。<**

编辑 dovecot 的配置文件(将第 25 行的注释符(#号)去掉):

[root@mail~] # vim /etc/dovecot/conf.d/10-mail.conf

mail_location = mbox:~/mail:INBOX=/var/mail/%u

**第 4 步:创建邮件的存储目录:**

[root@mail~] # su - boss

Last login: Sat Aug 15 16:15:58 CST 2015 on pts/1

[boss@mail ~]$ mkdir -p mail/.imap/INBOX

**第 5 步:启动 Dovecot 服务程序:**

[root@mail~] *# systemctl restart dovecot*

[root@mail~] *# systemctl enable dovecot*

ln -s '/usr/lib/systemd/system/dovecot.service' '/etc/systemd/system/multi-user.target.wants/dovecot.service'

### 14.2.3 用户使用邮局系统

您可以在《**软件资源库**》下载到 Windows7 系统以及 OutLook2007 邮件管理工具,系统网卡请按要求配置 IP 地址:

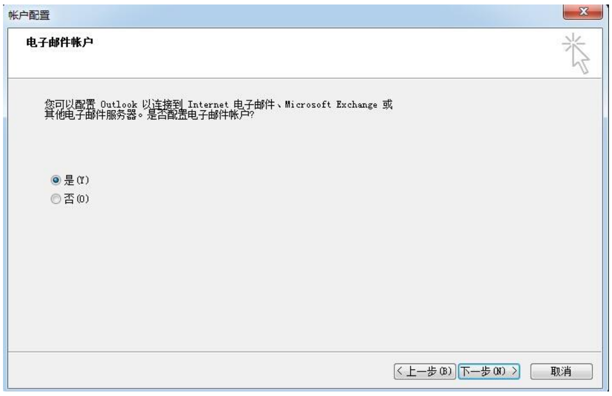
| 主机名称 | 操作系统 | IP 地址 |
| --- | --- | --- |
| 邮局服务器 | 红帽 RHEL7 操作系统 | 192.168.10.10 |
| DNS 服务器 | 红帽 RHEL7 操作系统 | 192.168.10.20 |
| 用户端主机 | 微软 Windows7 系统 | |

在 outlook 中登陆 boss 用户后尝试给 root@linuxprobe.com 发送邮件。

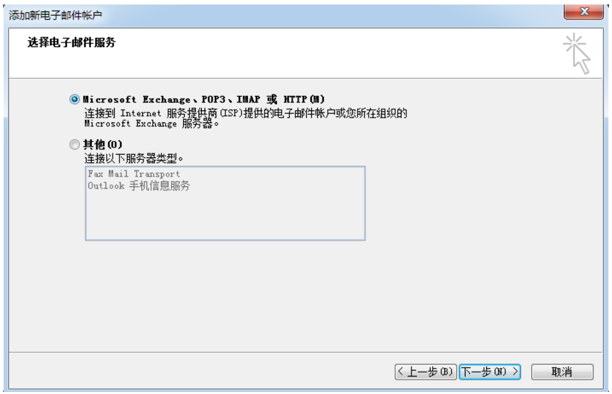
第 1 步：开启 OutLook 程序。



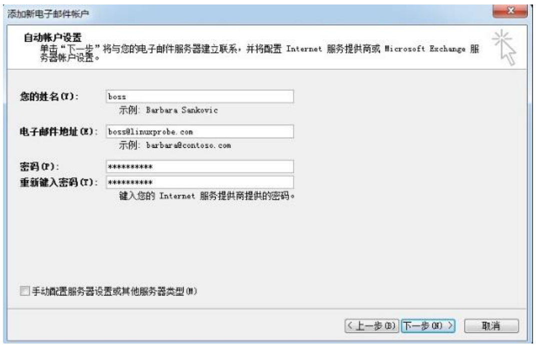
第 2 步：选择开始配置电子邮件帐户。



第 3 步：选择默认的邮局服务器类型。



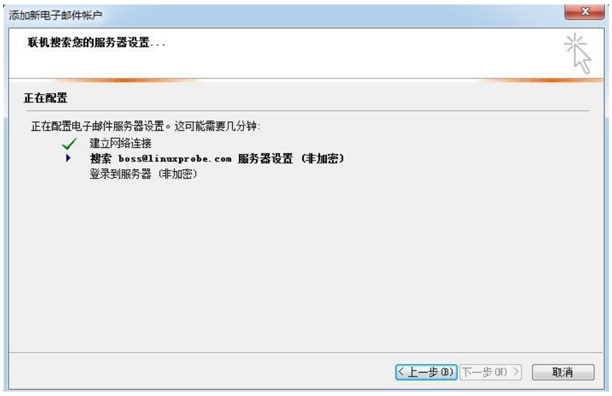
第 4 步：填写创建的邮箱帐号和密码。



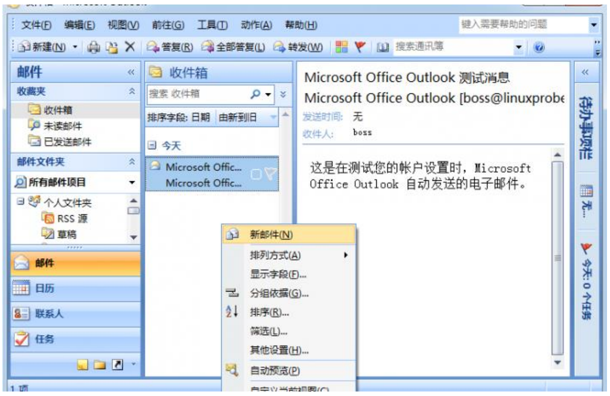
第 5 步：等待连接邮局



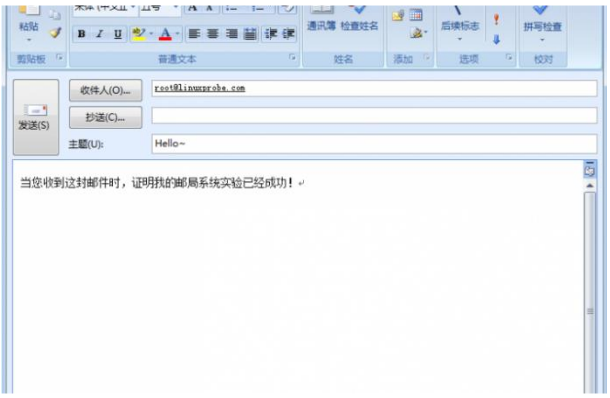

第 6 步：选择非加密的链接方式。




第 7 步：在页面上邮件，选择“新邮件”。




第 8 步：填写收件人与邮件内容后发送。



登陆到邮局服务器(192.168.10.10)后查看 root 用户的邮件：

[root@mail~] # **mail**

Heirloom Mail version 12.5 7/5/10.Type ? for help.

"/var/mail/root": 3 messages 3 unread >

U 1 user@localhost.com Fri Jul 10 09:58 1631/123113 "[abrt] full crash r"

U 2 Anacron Sat Aug 15 13:33 18/624 "Anacron job 'cron.dai"

U 3 boss Sat Aug 15 19:02 118/3604 "Hello~"

&> **3**

Message 3:

From boss@linuxprobe.com Sat Aug 15 19:02:06 2015

Return-Path:

X-Original-To: root@linuxprobe.com

Delivered-To: root@linuxprobe.com

From: "boss"

To:

Subject: Hello~

Date: Sat, 15 Aug 2015 19:02:06 +0800

Content-Type: text/plain; charset="gb2312"

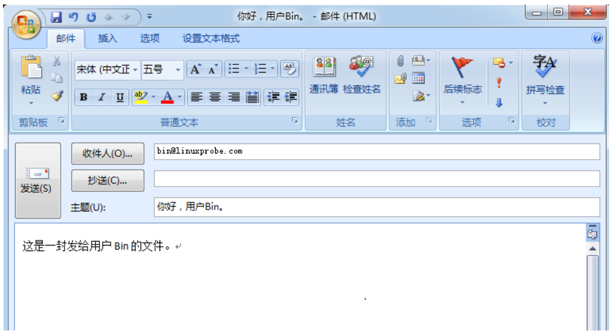
当您收到这封邮件时，证明我的邮局系统实验已经成功！

> **quit**

Held 3 messages in /var/mail/root

### 14.3 设置用户别名邮箱

我们刚刚顺利的向 root 用户发送了邮件，再来试试向 bin 用户（系统默认创建）的用户发送一封邮件吧：




尝试切换到 bin 用户（提示此帐户当前不可用）：

[root@mail ~]# su bin

This account is currently not available.

查看 root 用户的邮箱：

[root@mail ~]# mail

Heirloom Mail version 12.5 7/5/10.

Type ? for help.

"/var/mail/root": 4 messages 4 new >

U 1 user@localhost.com Fri Jul 10 09:58 1630/123103 "[abrt] full crash r"

U 2 Anacron Wed Aug 19 17:47 17/619 "Anacron job 'cron.dai"

U 3 boss Sat Aug 15 19:02 118/3604 "Hello~" U

4 boss Wed Aug 19 18:49 116/3231 "你好，用户 Bin。"

&> 4
Message 4:
From boss@linuxprobe.com Wed Aug 19 18:49:05 2015
Return-Path: <boss@linuxprobe.com>
X-Original-To: bin@linuxprobe.com
Delivered-To: bin@linuxprobe.com
From: "boss" <boss@linuxprobe.com>
To: <bin@linuxprobe.com>
Subject: 你好，用户 Bin。
Date: Wed, 19 Aug 2015 18:49:05 +0800
Content-Type: multipart/alternative; boundary="----=_NextPart_000_0006_01D0DAAF.B9104E90"
X-Mailer: Microsoft Office Outlook 12.0 Thread-Index: AdDabKrQzUHVBTgRQMaCtUsVtqfL1Q== Content-Language: zh-cn Status: R Content-Type: text/plain; charset="gb2312"
这是一封发给用户 Bin 的文件。
&> exit

好奇怪！这封明明发送给 bin 用户的邮件为什么会被 root 收到呢？这就是 aliases 别名机制！

查看 aliases 别名机制的配置文件：

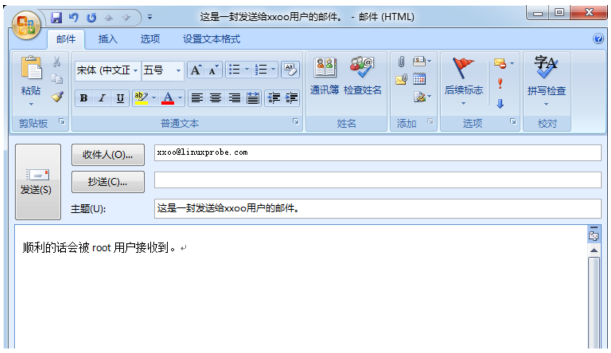
[root@mail ~]# cat /etc/aliases
#
# Aliases in this file will NOT be expanded in the header from
# Mail, but WILL be visible over networks or from /bin/mail.
#
# >>>>>>>>>> The program "newaliases" must be run after
# >> NOTE >> this file is updated for any changes to
# >>>>>>>>>> show through to sendmail.
#
# Basic system aliases -- these MUST be present.
mailer-daemon: postmaster
postmaster: root
# General redirections for pseudo accounts.

这样看完好像就大致明白了吧，原来这个文件定义了用户名与它的别名，格式为“**别名 用户名**”。

如果我们希望所有发送给 xxoo@linuxprobe.com 的邮件，均保存到 root@linuxprobe.com 的邮箱中，则这样追加：

# General redirections for pseudo accounts.
xxoo: root

顺利编辑/etc/aliases 文件后需要执行命令”**newaliases**“，这样追加的用户别名才能立即生效，然后尝试发送邮件。



[root@mail ~]# mail
Heirloom Mail version 12.5 7/5/10. Type ? for help.
"/var/mail/root": 5 messages 1 new 4 unread
U 1 user@localhost.com Fri Jul 10 09:58 1631/123113 "[abrt] full crash report"
U 2 Anacron Wed Aug 19 17:47 18/629 "Anacron job 'cron.daily' on mail.linuxprobe.com"
U 3 boss Wed Aug 19 18:44 114/2975 "hello"
4 boss Wed Aug 19 18:49 117/3242 "你好，用户 Bin。"
>N 5 boss Wed Aug 19 19:18 115/3254 "这是一封发送给 xxoo 用户的邮件。"

而如果想取消别名的话，只需编辑掉 **aliases** 文件中的用户别名后执行 **newaliases** 命令即可，很简单吧~

**本章结束，您可以在此写下笔记：**

第 15 章 使用 Squid 部署代理缓存服务。

**章节概述：**

本章节从代理缓存服务的工作原理开始讲起，让读者能够清晰理解**正向代理**（普通模式、透明模式）与**反向代理**的作用。正确的使用 Squid 服务程序部署代理缓存服务可以有效提升访问静态资源的效率，降低原服务器的负载。

不仅如此，还为读者们添加了对指定 IP 地址、网页关键词、网址与文件后缀的 ACL 访问限制功能的实验，真的很实用哦~

## 15.1 代理缓存服务

Squid 服务程序是一款在类 Unix 系统中最为流行的高性能代理服务软件，通常会被当作网站的前置缓存服务，用于替代用户向网站服务器请求页面数据并进行缓存，通俗来讲，Squid 服务程序会接收用户的请求，然后自动去下载指定数据（如网页）并存储在服务器内，当以后的用户再来请求相同数据时，则直接将刚刚储存在服务器本地的数据交给用户，减少了用户的等待时间。

Squid 服务程序配置起来相对简单，效率高、支持如 HTTP、FTP、SSL 等多种协议的数据缓存，还支持基于 ACL 访问控制列表和 ARL 访问权限列表功能的内容过滤与权限管理功能，禁止用户访问存在威胁或不适宜的网站资源，保证内网安全的同时还整体的提高了客户机的访问速度，帮助节省网络带宽，尤其适合安装在内存大、硬盘转速快的服务器上。



**从作用上分为正向代理和反向代理：**

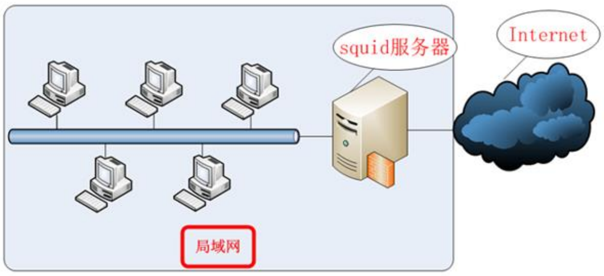
正向代理让用户可以通过 Squid 服务程序获取网站页面等数据，具体工作形式又分为标准代理模式与透明代理模式。

标准正向代理模式：

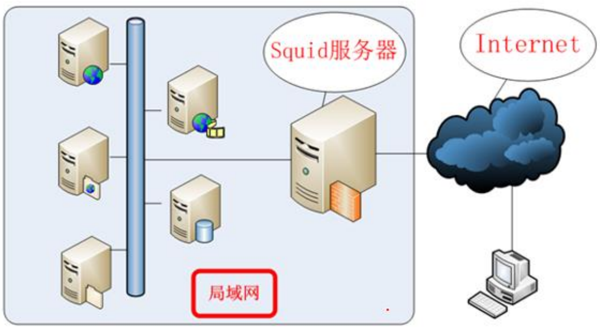
将网站的数据缓存在服务器本地，提高数据资源被再次访问时的效率，但用户必需在上网时指定代理服务器的 IP 地址与端口号，否则将不使用 Squid 服务。

透明正向代理模式：

功能作用与标准正向代理模式完全相同，但用户不需要指定代理服务器的 IP 地址与端口号，所以这种代理服务对于用户来讲是完全透明的。



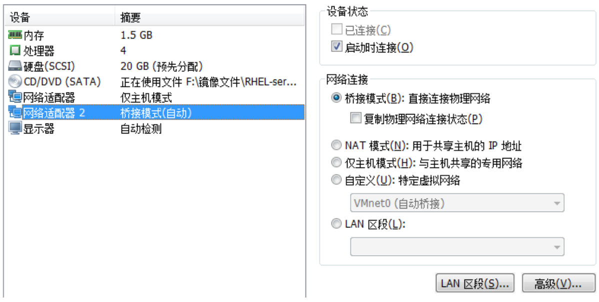

反向代理则是为了降低网站服务器负载而设计的，反向代理服务器负责回应用户对原始网站服务器的静态页面请求，即如果反向代理服务器中正巧有用户要访问的静态资源则直接将缓存的内容发送给用户，减少了对原始服务器的部分数据资源请求。



所以对于正向代理一般用于企业的局域网内，让员工通过 Squid 服务程序来代理上网，不但能节省网络带宽资源还能限制访问的页面，而反向代理则大多搭建在网站架构中，用于缓存网站的静态数据（如图片、HTML 静态网页、JS、CSS 框架文件等）。

### 15.2 配置 Squid 服务程序

本小节将为大家演示如何部署 Squid 服务的正向代理与反向代理，首先我们需要再添加一块网卡设备（桥接模式）：




按照下面的表单配置 IP 地址：

| 主机名称 | 操作系统 | IP 地址 |
|---|---|---|
| 服务端 | 红帽 RHEL7 操作系统 | 外网卡:桥接 DHCP 模式<br>内网卡:192.168.10.10 |
| 用户端 | 微软 Windows7 操作系统 | 192.168.10.20 |

测试是否能够访问互联网：

```
[root@linuxprobe Desktop]# ping www.linuxprobe.com
PING www.linuxprobe.com (162.159.211.33) 56(84) bytes of data.
64 bytes from 162.159.211.33: icmp_seq=1 ttl=45 time=166 ms
64 bytes from 162.159.211.33: icmp_seq=2 ttl=45 time=168 ms
64 bytes from 162.159.211.33: icmp_seq=3 ttl=45 time=167 ms
64 bytes from 162.159.211.33: icmp_seq=4 ttl=45 time=166 ms
^C
--- www.linuxprobe.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 166.361/167.039/168.109/0.836 ms
```

安装 squid 服务程序：

```
[root@linuxprobe ~]# yum install squid
Loaded plugins: langpacks, product-id, subscription-manager
………………省略部分安装过程………………
Installing:
 squid          x86_64      7:3.3.8-11.el7      rhel7      2.6 M
………………省略部分安装过程………………
Complete!
```

先别着急配置哦!我们先来了解下 squid 服务程序的主要文件吧:

| 主服务程序 | /usr/sbin/squid |
| --- | --- |
| 配置文件目录 | /etc/squid |
| 主配置文件 | /etc/squid/squid.conf |
| 访问日志文件 | /var/log/squid/access.log |
| 缓存日志文件 | /var/log/squid/cache.log |

而 squid 服务程序中常用的参数有:

| 参数 | 作用 |
| --- | --- |
| http_port 3128 | 监听的端口号。 |
| cache_mem 64M | 内存缓冲区的大小。 |
| cache_dir ufs /var/spool/squid 2000 16 256 | 硬盘缓冲区的大小。 |
| cache_effective_user squid | 设置缓存的有效用户。 |
| cache_effective_group squid | 设置缓存的有效用户组。 |
| dns_nameservers IP 地址 | 一般不设置，用服务器默认的 DNS 地址。 |
| cache_access_log /var/log/squid/access.log | 访问日志文件的保存路径。 |
| cache_log /var/log/squid/cache.log | 缓存日志文件的保存路径。 |
| visible_hostname linuxprobe.com | 设置 Squid 服务主机的名称。 |

启动 squid 服务程序并加入到开机启动项:

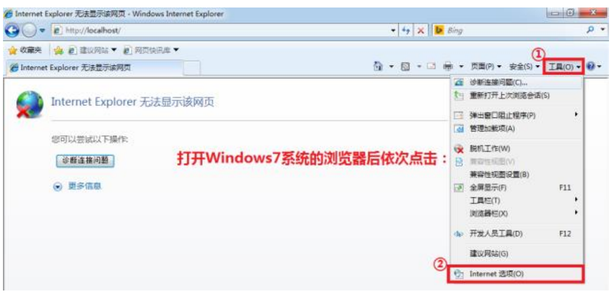
[root@linuxprobe ~]# *systemctl restart squid*

[root@linuxprobe ~]# *systemctl enable squid*

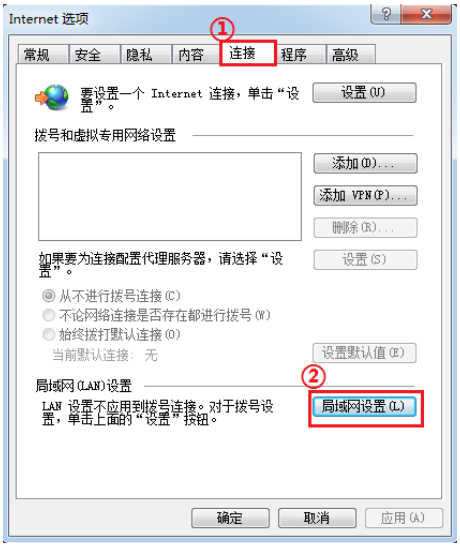
ln -s '/usr/lib/systemd/system/squid.service' '/etc/systemd/system/multi-user.target.wants/squid.service'

### 15.2.1 标准正向代理

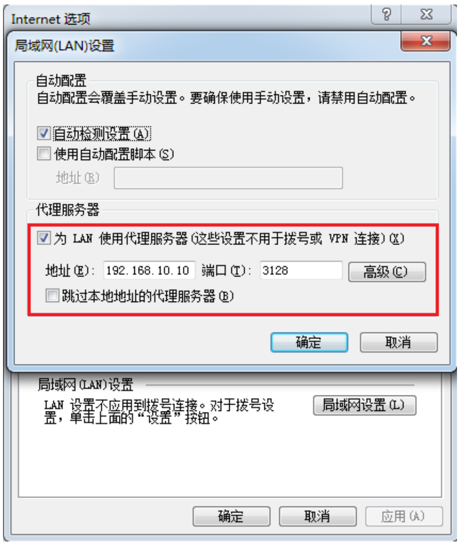
当 Squid 服务程序顺利启动后，默认即可使用标准正向代理模式啦，在 Windows7 系统中打开浏览器后依次点击:



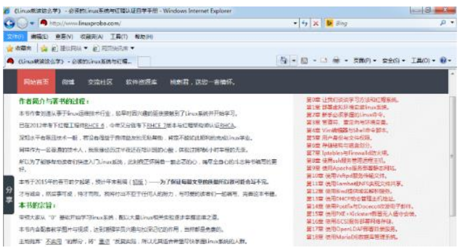
在网络选项中点击“连接”→“局域网设置”：




填写 Squid 服务器的 IP 地址与端口号




尝试访问网站：



Squid 服务程序默认会占用 3128、3401 与 4827 端口，我们也可以将端口号修改为其他的哦， 编辑配置文件（**修改第 59 行**）：

[root@linuxprobe ~]# vim /etc/squid/squid.conf

http_port 10000

使用 setsebool 命令来限制 squild 服务只能使用自定义的端口号：

[root@linuxprobe ~]# setsebool -P squid_connect_any 0

查看当前 SElinux 允许的服务端口：

[root@linuxprobe ~]#   semanage port -l | grep -w -i squid_port_t

| | | |
|---|---|---|
| squid_port_t | tcp | 3128, 3401, 4827 |
| squid_port_t | udp | 3401, 4827 |

添加 SELinux 对 10000 端口的允许策略：

[root@linuxprobe ~]# semanage port -a -t squid_port_t -p tcp 10000

[root@linuxprobe ~]# semanage port -l | grep -w -i squid_port_t

squid_port_t tcp **10000**, 3128, 3401, 4827

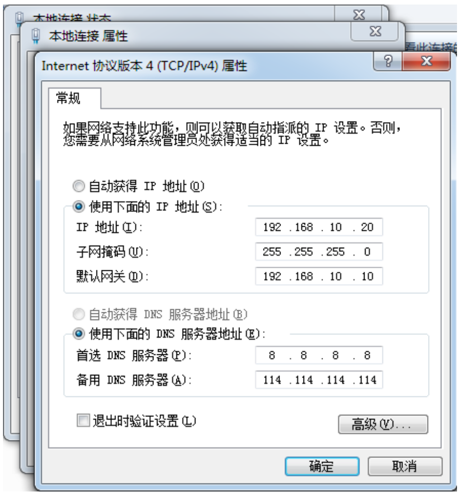
squid_port_t udp 3401, 4827

重启 squid 服务程序后即可生效：

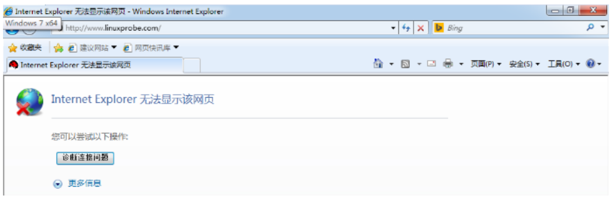
[root@linuxprobe ~]# systemctl restart squid

### 15.2.2 透明正向代理

如果要想实现透明正向代理，则必需将用户的网关 IP 指向 Squid 服务器，而此后便无需再修改浏览器选项：




尝试访问网站失败：




在命令提示符中 ping 下域名：

C:\Users\linuxprobe>ping www.linuxprobe.com

Ping 请求找不到主机 www.linuxprobe.com。请检查该名称，然后重试。

原来 Squid 服务程序是不支持 DNS 解析代理的，这个就需要配置 SNAT 啦。如果忘记 SNAT 技术了，没关系，回去再看下防火墙的章节吧，-o 参数后面写外网出口的网卡名称：

[root@linuxprobe ~]# iptables -t nat -A POSTROUTING -p udp --dport 53 -o eno33554968 -j MASQUERADE

开启 Ipv4 的转发策略：

[root@linuxprobe ~]# echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf

[root@linuxprobe ~]# sysctl -p

net.ipv4.ip_forward = 1

再次尝试 Ping 下网站域名：

C:\Users\linuxprobe>ping www.linuxprobe.com

正在 Ping www.linuxprobe.com [116.31.127.233] 具有 32 字节的数据:

116.31.127.233 的 Ping 统计信息:

数据包: 已发送 = 4，已接收 = 0，丢失 = 4 (100% 丢失)，

不错哦~现在 DNS 已经能够正常工作啦，来配置透明正向代理吧，编辑配置文件：

[root@linuxprobe ~]# **vim /etc/squid/squid.conf**

//在第 59 行后面添加参数 transparent

**http_port 3128 transparent**

判断配置文件是否有错误(会有很多输出值)：

[root@linuxprobe ~]# squid -k parse

重新启动 squid 服务程序：

[root@linuxprobe ~]# systemctl restart squid

将用户对 80 端口的请求转发至 3128 端口：

[root@linuxprobe ~]# *iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 3128*

[root@linuxprobe ~]# *iptables -t nat -R PREROUTING 1 -i eno33554968 -p tcp --dport 80 -j REDIRECT --to-port 3128*

快去客户机尝试访问网站吧：



### 15.2.2 反向代理

反向代理的作用是将网站中的静态资源本地化，也就是将一部分本应该由原始服务器处理的请求交给 Squid 缓存服务处理。

编辑 Squid 服务程序的配置文件（正向代理与反向代理不能同时使用，请还原您前面修改过的参数）：

[root@linuxprobe ~]# **vim /etc/squid/squid.conf**

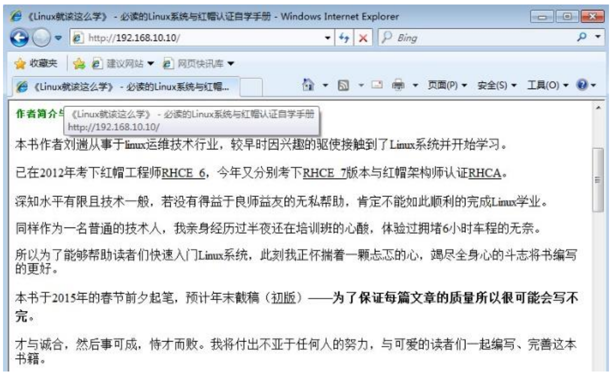
//第 59 行，修改格式为：http_port Squid 服务器地址:监听端口号 vhost

http_port 192.168.10.10:80 vhost

//第 60 行，添加格式为：cache_peer 原网站服务器地址 parent 服务器端口号 0 originserver

cache_peer 106.184.1.125 parent 80 0 originserver

然后使用客户机去访问网站：




因为《Linux 就该这么学》的网站框架使用了动态资源，所以现在看起来会有些乱乱的。

### 15.3 ACL 访问控制

Squid 服务的 ACL 访问控制是非常有用的功能，可以根据特定条件来进行数据缓存或限制用户的访问，ACL 元素的定义语法为：

acl aclname acltype string

acl aclname acltype "file"

src 定义来源地址（即用户的客户机 IP 地址）：

acl aclname src ip-address/netmask

acl aclname src addr1-addr2/netmask

dst 定义目标地址（即用户请求的网站 IP 地址）：

acl aclname dst ip-address/netmask

port 用于指定访问端口：

acl aclname port 80 1024

acl aclname port 0-1024

url_regex 用于限制网址中的关键词：

acl aclname url_regex [-i] pattern

proto 用于定义要代理的协议：

acl aclname proto HTTP FTP

method 用于指定请求的方法：

acl aclname method GET POST

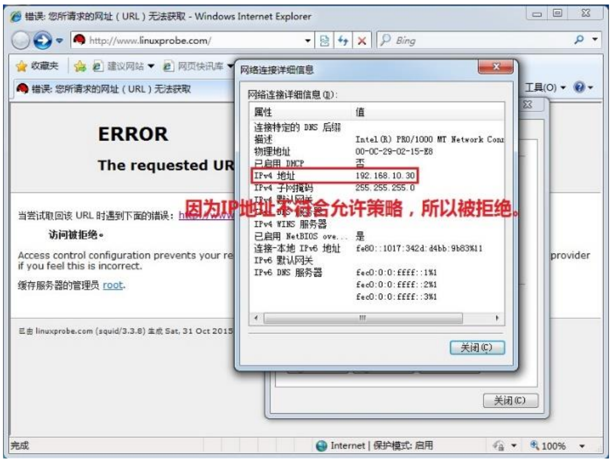
访问控制列表由多个规则条目组成的，根据指定的条件来允许或限制访问请求，匹配顺序会由上至下，一旦匹配则立即结束，通常会在控制列表的最下面写上“deny all”或者“allow all”来避免安全隐患。

仅允许 192.168.10.20 的主机使用本地 Squid 服务，拒绝其余主机：

acl client src 192.168.10.20

http_access allow client

http_access deny all



拒绝客户机使用代理服务器访问带有关键词“linux”的网站：

acl deny_keyword url_regex -i linux

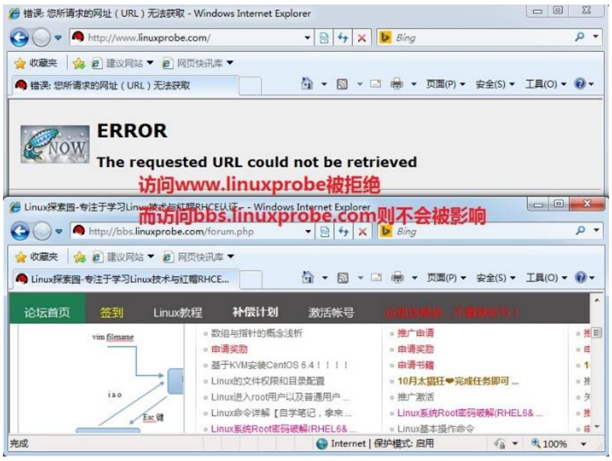
http_access deny deny_keyword




拒绝客户机使用代理服务器访问《Linux 就该这么学》的网站：

acl deny_url url_regex http://www.linuxprobe.com

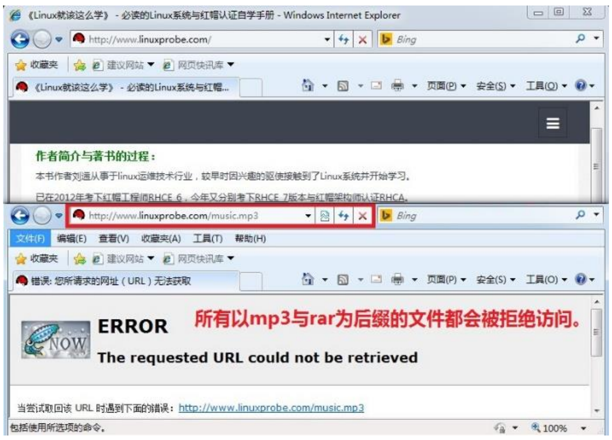
http_access deny deny_url




禁止客户机使用代理服务器下载以 mp3 与 rar 为后缀的文件：

acl badfile urlpath_regex -i \.mp3$ \.rar$

http_access deny badfile



第 16 章 使用 iSCSI 服务部署网络存储。

**章节概述：**

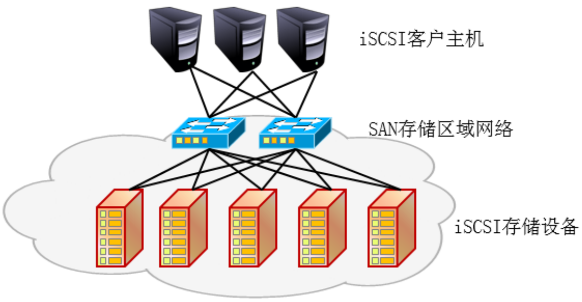
本章节将分析 SCSI 与 iSCSI 技术结构的不同，了解 iSCSI 技术的优势、SAN 存储网络技术结构以及 iSCSI HBA 卡的作用。
完整演示部署 iSCSI target 服务程序的方法流程：创建 RAID 阵列(5)后使用 targetcli 命令发布到 iSCSI 存储目录并创建 ACL 列表。
配置使用 iSCSI initiator 服务程序发现、连接并使用 iSCSI 存储设备，最后编辑 fstab 文件将存储设备设置为开机启动。

## 16.1 网络存储技术

传统的 **SCSI 小型计算机系统接口**(Small Computer System Interface)技术是存储设备最基本的标准协议，但通常需要设备互相靠近并用 SCSI 总线链接，因此受到了物理环境的限制。

**iSCSI 小型计算机系统接口**（即 Internet Small Computer System Interface）则是由 IBM 公司研究开发用于实现在 IP 网络上运行 SCSI 协议的新存储技术，即能够让 SCSI 接口与以太网技术相结合，使用 iSCSI 协议基于以太网传送 SCSI 命令与数据，克服了 SCSI 需要直接连接存储设备的局限性，使得我们可以跨越不同的服务器共享存储设备，并可以做到不停机状态下扩展存储容量。

**SAN 存储区域网络技术**(Storage Area Network)便是基于 iSCSI 存储协议，采用高速光钎通道传输存储数据的服务程序。




**本图为 SAN 结构拓扑**

服务器会基于 iSCSI 协议将 SCSI 设备、命令与数据打包成标准的 TCP/IP 包然后通过 IP 网络传输到目标存储设备，而远端存储设备接收到数据包后需要基于 iSCSI 协议将 TCP/IP 包解包成 SCSI 设备、命令与数据，这个过程无疑会消耗系统 CPU 资源，因此我们可以将 SCSI 协议的封装动作交由独立的 iSCSI HBA 硬件卡来处理，减少了对服务器性能的影响。



**本图中设备为 iSCSI HBA 卡**

但坦白来讲 iSCSI 技术还是存在诸多问题的，如距离与带宽之间的矛盾关系，虽然 iSCSI 满足了数据长距离传输的需求，但现在广域网的带宽还是不够理想，IP 网络的速率和延迟都是 iSCSI 传输数据的巨大障碍。

## 16.2 部署 iSCSI 存储

**iSCSI 的工作方式分为服务端（target）与客户端（initiator）：**

　　服务端：即存放硬盘或 RAID 设备的存储端，目的是为客户端提供可用的存储。

　　客户端：使用服务端的服务器主机。

**本实验需要两台虚拟主机来完成，分别是：**

| 主机名称 | 操作系统 | IP 地址 |
| --- | --- | --- |
| iscsi 服务端 | 红帽 RHEL7 操作系统 | 192.168.10.10 |
| iscsi 客户端 | 红帽 RHEL7 操作系统 | 192.168.10.20 |

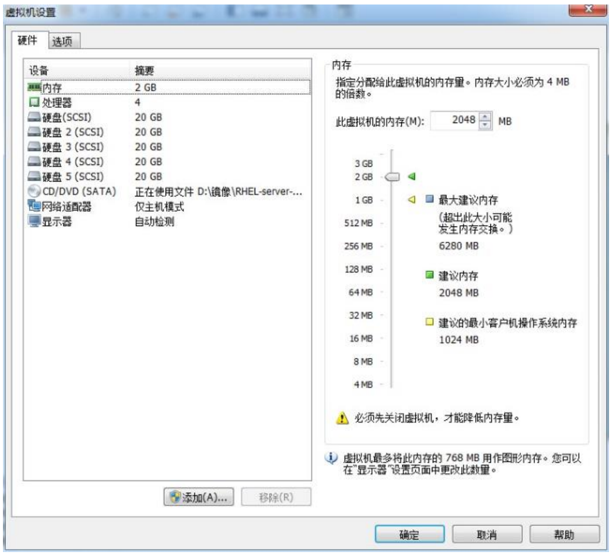
**逻辑单元** LUN(即 Logical Unit Number)是使用 iSCSI 协议中的重要概念，因为当客户机想要使用服务端存储设备时都必需输入对应的名称(Target ID)，而一个服务端可能会同时提供多个可用的存储设备，于是便用 LUN 来详细的描述设备或对象，同时每个 LUN Device 可能代表一个硬盘或 RAID 设备，LUN 的名称由用户指定。

### 16.2.1 配置 iSCSI 服务端

**第 1 步:准备作为 LUN 发布的存储设备。**

在前面的存储结构章节中学习了使用 mdadm 工具创建 RAID 磁盘冗余阵列的方法，忘记就翻回去看下吧~

在虚拟机中再添加 4 块硬盘：




创建 RAID5 并设置 1 块备份故障盘：

```
[root@linuxprobe ~]# mdadm -Cv /dev/md0 -n 3 -l 5 -x 1 /dev/sdb /dev/sdc /dev/sdd /dev/sde
mdadm: layout defaults to left-symmetric
mdadm: layout defaults to left-symmetric
mdadm: chunk size defaults to 512K
mdadm: size set to 20954624K
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
```

查看 RAID 阵列的详细信息，记录下 UUID 的值：

```
[root@linuxprobe ~]# mdadm -D /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Thu Sep 24 21:59:57 2015
     Raid Level : raid5
     Array Size : 41909248 (39.97 GiB 42.92 GB)
  Used Dev Size : 20954624 (19.98 GiB 21.46 GB)
```

```
     Raid Devices : 3
    Total Devices : 4
      Persistence : Superblock is persistent
      Update Time : Thu Sep 24 22:02:23 2015
            State : clean
   Active Devices : 3
  Working Devices : 4
   Failed Devices : 0
    Spare Devices : 1
           Layout : left-symmetric
       Chunk Size : 512K
             Name : linuxprobe.com:0  (local to host linuxprobe.com)
             UUID : 3370f643:c10efd6a:44e91f2a:20c71f3e
           Events : 26
    Number   Major   Minor   RaidDevice State
       0       8       16        0      active sync   /dev/sdb
       1       8       32        1      active sync   /dev/sdc
       4       8       48        2      active sync   /dev/sdd
       3       8       64        -      spare   /dev/sde
```

创建 RAID 阵列配置文件:

```
[root@linuxprobe ~]# vim /etc/mdadm.conf
DEVICE /dev/sdb /dev/sdc /dev/sdd /dev/sde
ARRAY /dev/md0 UUID=3370f643:c10efd6a:44e91f2a:20c71f3e
```

**第 2 步:安装 iSCSI target 服务程序:**

```
[root@linuxprobe ~]# yum -y install targetd targetcli
Loaded plugins: langpacks, product-id, subscription-manager
………………省略部分安装信息………………
Installing:
 targetcli          noarch      2.1.fb34-1.el7        rhel7      55 k
 targetd            noarch      0.7.1-1.el7           rhel7      48 k
Complete!
```

启动 iSCSI target 服务程序:

```
[root@linuxprobe ~]# systemctl start targetd
```

将 iSCSI target 服务程序添加到开机启动项:

```
[root@linuxprobe ~]# systemctl enable targetd
ln -s '/usr/lib/systemd/system/targetd.service' '/etc/systemd/system/multi-user.target.wants/targetd.service'
```

**第 3 步:创建存储对象。**

**targetcli 命令用于管理 iSCSI target 存储设备，格式为：“targetcli”**

```
[root@linuxprobe ~]# targetcli
Warning: Could not load preferences file /root/.targetcli/prefs.bin.
targetcli shell version 2.1.fb34
Copyright 2011-2013 by Datera, Inc and others.
For help on commands, type 'help'.
```

查看当前的存储目录树：

```
/> ls
o- / ...................................................................... [...]
o- backstores .......................................................... [...]
| o- block .............................................. [Storage Objects: 0]
| o- fileio ............................................. [Storage Objects: 0]
| o- pscsi .............................................. [Storage Objects: 0]
| o- ramdisk ............................................ [Storage Objects: 0]
o- iscsi ........................................................ [Targets: 0]
o- loopback ..................................................... [Targets: 0]
```

进入/backstores/block 目录中：

```
/> cd /backstores/block
/backstores/block>
```

使用/dev/md0 创建设备 disk0：

```
/backstores/block> create disk0 /dev/md0
Created block storage object disk0 using /dev/md0.
```

返回到根目录中：

```
/backstores/block> cd ..
/backstores> cd ..
/>
```

查看创建后的设备：

```
/> ls
o- / ...................................................................... [...]
  o- backstores .......................................................... [...]
  | o- block .............................................. [Storage Objects: 1]
  | | o- disk0 ..................... [/dev/md0 (40.0GiB) write-thru deactivated]
  | o- fileio ............................................. [Storage Objects: 0]
  | o- pscsi .............................................. [Storage Objects: 0]
  | o- ramdisk ............................................ [Storage Objects: 0]
  o- iscsi ........................................................ [Targets: 0]
  o- loopback ..................................................... [Targets: 0]
```

**第 4 步:配置 iSCSI target 目标。**

进入到 iscsi 目录中：

```
/> cd iscsi
/iscsi>
```

创建 iSCSI target 目标：

```
/iscsi> create
Created target iqn.2003-01.org.linux-iscsi.linuxprobe.x8664:sn.d497c356ad80.
Created TPG 1.
```

依次进入到 target 的 luns 目录中：

```
/iscsi> cd iqn.2003-01.org.linux-iscsi.linuxprobe.x8664:sn.d497c356ad80/
/iscsi/iqn.20....d497c356ad80> ls
o- iqn.2003-01.org.linux-iscsi.linuxprobe.x8664:sn.d497c356ad80 ...... [TPGs: 1]
  o- tpg1 .............................................. [no-gen-acls, no-auth]
```

```
    o- acls .......................................................... [ACLs: 0]
    o- luns .......................................................... [LUNs: 0]
    o- portals ..................................................... [Portals: 0]
/iscsi/iqn.20....d497c356ad80> cd tpg1/
/iscsi/iqn.20...c356ad80/tpg1> cd luns
/iscsi/iqn.20...d80/tpg1/luns>
```

创建 LUN 设备：

```
/iscsi/iqn.20...d80/tpg1/luns> create /backstores/block/disk0
Created LUN 0.
```

**第 5 步：设置访问控制列表。**

切换到 acls 目录中：

```
/iscsi/iqn.20...d80/tpg1/luns> cd ..
/iscsi/iqn.20...c356ad80/tpg1> cd acls
```

创建访问控制列表：

```
/iscsi/iqn.20...d80/tpg1/acls> create iqn.2003-01.org.linux-iscsi.linuxprobe.x8664:sn.d497c356ad80:client
Created Node ACL for iqn.2003-01.org.linux-iscsi.linuxprobe.x8664:sn.d497c356ad80:client
Created mapped LUN 0.
```

切换到 portals 目录中：

```
/iscsi/iqn.20...d80/tpg1/acls> cd ..
/iscsi/iqn.20...c356ad80/tpg1> cd portals
```

添加允许监听的 IP 地址：

```
/iscsi/iqn.20.../tpg1/portals> create 192.168.10.10
Using default IP port 3260
Created network portal 192.168.10.10:3260.
```

查看配置概述后退出工具：

```
/iscsi/iqn.20.../tpg1/portals> ls /
o- / ............................ [...]
  o- backstores................. [...]
  | o- block ................... [Storage Objects: 1]
  | | o- disk0 ................. [/dev/md0 (40.0GiB) write-thru activated]
  | o- fileio .................. [Storage Objects: 0]
  | o- pscsi ................... [Storage Objects: 0]
  | o- ramdisk ................. [Storage Objects: 0]
  o- iscsi ..................... [Targets: 1]
  | o- iqn.2003-01.org.linux-iscsi.linuxprobe.x8664:sn.d497c356ad80 .... [TPGs: 1]
  |     o- tpg1 ................. [no-gen-acls, no-auth]
  |        o- acls ....................................................... [ACLs: 1]
  |        | o- iqn.2003-01.org.linux-iscsi.linuxprobe.x8664:sn.d497c356ad80:client [Mapped LUNs: 1]
  |        |     o- mapped_lun0 ............................................ [lun0 block/disk0 (rw)]
    o- luns .................... [LUNs: 1]
  |        | o- lun0 ............. [block/disk0 (/dev/md0)]
  |        o- portals ............ [Portals: 1]
  |            o- 192.168.10.20:3260    [OK]
  o- loopback .................. [Targets: 0]
```

/> exit
Global pref auto_save_on_exit=true
Last 10 configs saved in /etc/target/backup.
Configuration saved to /etc/target/saveconfig.json

第 4 步:创建防火墙允许规则：

[root@linuxprobe ~]# firewall-cmd --permanent  --add-port=3260/tcp
success
[root@linuxprobe ~]# firewall-cmd --reload
success

**16.2.2 配置 iSCSI 客户端**

首先检查能够与 iscsi 服务端通信：

[root@linuxprobe ~]# ping -c 4 192.168.10.10
PING 192.168.10.10 (192.168.10.10) 56(84) bytes of data.
64 bytes from 192.168.10.10: icmp_seq=1 ttl=64 time=0.959 ms
64 bytes from 192.168.10.10: icmp_seq=2 ttl=64 time=0.469 ms
64 bytes from 192.168.10.10: icmp_seq=3 ttl=64 time=0.465 ms
64 bytes from 192.168.10.10: icmp_seq=4 ttl=64 time=0.277 ms

--- 192.168.10.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 0.277/0.542/0.959/0.253 ms

红帽 RHEL7 系统已经默认安装了 iscsi 客户端服务程序：

[root@linuxprobe ~]# yum install iscsi-initiator-utils
Loaded plugins: langpacks, product-id, subscription-manager
Package iscsi-initiator-utils-6.2.0.873-21.el7.x86_64 already installed and latest version
Nothing to do

**编辑的 iscsi 客户端名称文件：**

该名称是 initiator 客户端的唯一标识，读者可以按照我的方法修改，也可以用 iscsi-iname 命令随机生成~都可以的。

[root@linuxprobe ~]# vim /etc/iscsi/initiatorname.iscsi
**InitiatorName=iqn.2003-01.org.linux-iscsi.linuxprobe.x8664:sn.d497c356ad80:client**

重启 iscsi 客户端服务程序：

[root@linuxprobe ~]# systemctl restart iscsid

将 iscsi 客户端服务程序添加到开机启动项中：

[root@linuxprobe ~]# *systemctl enable iscsid*
ln -s '/usr/lib/systemd/system/iscsid.service' '/etc/systemd/system/multi-user.target.wants/iscsid.service'

发现 iscsi 服务端的可用存储设备：

**iscsiadm 命令用于管理（插入、查询、更新或删除）iSCSI 数据库配置文件的命令行工具，格式见下面演示。**

[root@linuxprobe ~]# iscsiadm -m discovery -t st -p 192.168.10.10
192.168.10.10:3260,1 iqn.2003-01.org.linux-iscsi.linuxprobe.x8664:sn.d497c356ad80

连接 iscsi 服务端的可用存储设备：

[root@linuxprobe ~]# *iscsiadm -m node -T iqn.2003-01.org.linux-iscsi.linuxprobe.x8664:sn.d497c356ad80 -p 192.168.10.10 --login*
Logging in to [iface: default, target: iqn.2003-01.org.linux-iscsi.linuxprobe.x8664:sn.d497c356ad80, portal: 192.168.10.10,3260] (multiple)

Login to [iface: default, target: iqn.2003-01.org.linux-iscsi.linuxprobe.x8664:sn.d497c356ad80, portal: 192.168.10.10,3260] successful.

此时便多了一块硬盘设备:

```
[root@linuxprobe ~]# file /dev/sdb
/dev/sdb: block special
```

格式化、挂载后查看容量信息:

```
[root@linuxprobe ~]# mkfs.xfs /dev/sdb
log stripe unit (524288 bytes) is too large (maximum is 256KiB)
log stripe unit adjusted to 32KiB
meta-data=/dev/sdb               isize=256    agcount=16, agsize=654720 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=0
data     =                       bsize=4096   blocks=10475520, imaxpct=25
         =                       sunit=128    swidth=256 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=0
log      =internal log           bsize=4096   blocks=5120, version=2
         =                       sectsz=512   sunit=8 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
[root@linuxprobe ~]# mkdir /iscsi
[root@linuxprobe ~]# mount /dev/sdb /iscsi
[root@linuxprobe ~]# df -h
Filesystem             Size  Used Avail Use% Mounted on
/dev/mapper/rhel-root   18G  3.4G   15G  20% /
devtmpfs               734M     0  734M   0% /dev
tmpfs                  742M  176K  742M   1% /dev/shm
tmpfs                  742M  8.8M  734M   2% /run
tmpfs                  742M     0  742M   0% /sys/fs/cgroup
/dev/sr0               3.5G  3.5G     0 100% /media/cdrom
/dev/sda1              497M  119M  379M  24% /boot
/dev/sdb                40G   33M   40G   1% /iscsi
```

查看设备的 UUID 值:

```
[root@linuxprobe ~]# blkid | grep /dev/sdb
/dev/sdb: UUID="eb9cbf2f-fce8-413a-b770-8b0f243e8ad6" TYPE="xfs"
```

设置为开机后自动挂载时因为 iSCSI 服务程序基于 IP 网络传输数据,所以我们必需在 fstab 文件中添加参数_netdev,代表网络联通后再挂载:

```
[root@linuxprobe ~]# vim /etc/fstab
UID=eb9cbf2f-fce8-413a-b770-8b0f243e8ad6 /iscsi xfs defaults,_netdev 0 0
```

# 第 17 章 使用 OpenLDAP 部署目录服务。

**章节概述：**

本章节将理解目录服务的概念，学习 OpenLdap 服务程序与 TLS 加密协议的部署方法，并自动挂载用户目录。

通过部署目录服务实现对系统帐户的集中式管理，相比于 X.500 协议具有更快的查询速度、更少的消耗资源等优势。

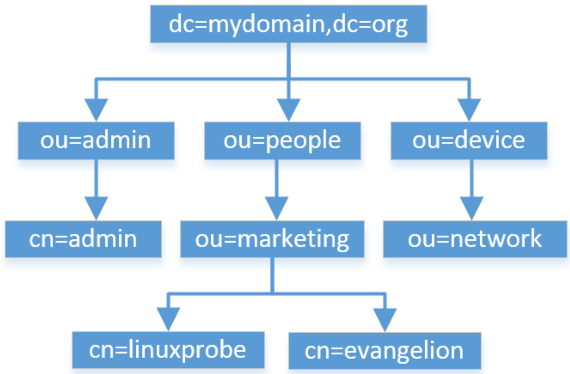
## 17.1 了解目录服务

回忆前面所学的章节，我们发现其实目录可以被理解成是一种为查询、浏览或搜索的数据库，但数据库又分为了目录数据库和关系数据库，目录数据库主要用于**存储较小的信息**（如姓名、电话、主机名等），同时**具有很好的读性能**，但在写性能方面比较差，所以不适合存放那些需要经常修改的数据。

**目录服务**则是由目录数据库和一套能够访问和处理数据库信息的协议组成的服务协议，用于集中的管理主机帐号密码，员工名字等数据，大大的提升了管理工作效率。

**轻量级目录访问协议** LDAP(Lightweight Directory Access Protocol)是在**目录访问协议 X.500** 的基础上研发的，主要的优势是：X.500 目录协议功能非常臃肿，消耗大量资源，无法做到快速查询且不支持 TCP/IP 协议网络。

LDAP 采用树状结构存储数据（类似于前面学习的 DNS 服务程序），用于在 IP 网络层面实现对分布式目录的访问和管理操作，**条目**是 LDAP 协议中最基本的元素，可以想象成字典中的单词或者数据库中的记录，通常对 LDAP 服务程序的添加、删除、更改、搜索都是以条目为基本对象的。




**LDAP 树状结构存储数据**

dn:每个条目的唯一标识符，如上图中 linuxprobe 的 dn 值是：

cn=linuxprobe,ou=marketing,ou=people,dc=mydomain,dc=org

rdn:一般为 dn 值中最左侧的部分，如上图中 linuxprobe 的 rdn 值是：cn=linuxprobe

base DN:此为基准 DN 值，表示顶层的根部，上图中的 base DN 值是：dc=mydomain,dc=org

而每个条目可以有多个属性（如姓名、地址、电话等），每个属性中会保存着对象名称与对应值，LDAP 已经为运维人员对常见的对象定义了属性，其中有：

| 属性名称 | 属性别名 | 语法 | 描述 | 值（举例） |
|---|---|---|---|---|
| commonName | cn | Directory String | 名子 | sean |
| surname | sn | Directory String | 姓氏 | Chow |
| organizationalUnitName | ou | Directory String | 单位（部门）名称 | IT_SECTION |
| organization | o | Directory String | 组织（公司）名称 | linuxprobe |
| telephoneNumber | | Telephone Number | 电话号码 | 911 |
| objectClass | | | 内置属性 | organizationalPerson |

## 17.2 目录服务实验

OpenLdap 是基于 LDAP 协议的开源程序，它的程序名称叫做 slapd，本次实验需要用到两台主机：

| 主机名称 | 操作系统 | IP 地址 |
|---|---|---|
| LDAP 服务端 (instructor.linuxprobe.com) | 红帽 RHEL7 操作系统 | 192.168.10.10 |
| LDAP 客户端 | 红帽 RHEL7 操作系统 | 192.168.10.20 |

### 17.2.1 配置 LDAP 服务端：

安装 openldap 与相关的软件包：

[root@linuxprobe ~]# **yum install -y openldap openldap-clients openldap-servers migrationtools**

Loaded plugins: langpacks, product-id, subscription-manager

………………省略部分安装过程………………

```
Installing:
 migrationtools      noarch     47-15.el7        rhel7      26 k
 openldap-clients    x86_64     2.4.39-3.el7     rhel7     183 k
 openldap-servers    x86_64     2.4.39-3.el7     rhel7     2.1 M
```

………………省略部分安装过程………………

Complete!

生成密钥文件（记下生成出的值，后面要用）：

[root@linuxprobe ~]# slappasswd -s linuxprobe -n > /etc/openldap/passwd

[root@linuxprobe ~]# cat /etc/openldap/passwd

**{SSHA}v/GJvGG8SbIuCxhfTDVhkmWEuz2afNIR**

写入一条主机与 IP 地址的解析记录：

[root@linuxprobe ~]# echo "192.168.10.10 instructor.linuxprobe.com" >> /etc/hosts

因为 LDAP 目录服务是以明文的方式在网络中传输数据的（包括密码），这样真的很不安全，所以我们采用 TLS 加密机制来解决这个问题，使用 openssl 工具生成 X509 格式的证书文件（有效期为 365 天）：

[root@linuxprobe ~]# *openssl req -new -x509 -nodes -out /etc/openldap/certs/cert.pem -keyout /etc/openldap/certs/priv.pem -days 365*

Generating a 2048 bit RSA private key

.........................................+++

...............................................................+++

writing new private key to '/etc/openldap/certs/priv.pem'

-----

You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

-----

Country Name (2 letter code) [XX]:敲击回车

State or Province Name (full name) []:敲击回车

Locality Name (eg, city) [Default City]:敲击回车

Organization Name (eg, company) [Default Company Ltd]:敲击回车
Organizational Unit Name (eg, section) []:敲击回车
Common Name (eg, your name or your server hostname) []:instructor.linuxprobe.com
修改证书的所属与权限：
[root@linuxprobe ~]# cd /etc/openldap/certs/
[root@linuxprobe certs]# chown ldap:ldap *
[root@linuxprobe certs]# chmod 600 priv.pem
[root@linuxprobe certs]# ls -al
total 8
drwxr-xr-x. 2 root root 36 Oct 5 13:41 .
drwxr-xr-x. 5 root root 100 Oct 5 13:39 ..
-rw-r--r--. 1 ldap ldap 1318 Oct 5 13:41 cert.pem
-rw-------. 1 ldap ldap 1704 Oct 5 13:41 priv.pem
复制一份 LDAP 的配置模板：
[root@linuxprobe ~]# cp /usr/share/openldap-servers/DB_CONFIG.example /var/lib/ldap/DB_CONFIG
生成数据库文件（不用担心报错信息）：
[root@linuxprobe ~]# slaptest
5610aaa9 hdb_db_open: database "dc=my-domain,dc=com": db_open(/var/lib/ldap/id2entry.bdb) failed: No such file or directory (2).
5610aaa9 backend_startup_one (type=hdb, suffix="dc=my-domain,dc=com"): bi_db_open failed! (2)
slap_startup failed (test would succeed using the -u switch)
修改 LDAP 数据库的所属主与组：
[root@linuxprobe ~]# chown ldap:ldap /var/lib/ldap/*
启动 slapd 服务程序并设置为开机启动：
[root@linuxprobe ~]# *systemctl restart slapd*
[root@linuxprobe ~]# *systemctl enable slapd*
ln -s '/usr/lib/systemd/system/slapd.service' '/etc/systemd/system/multi-user.target.wants/slapd.service'
在 LDAP 目录服务中使用 **LDIF**(LDAP Interchange Format)格式来保存信息，而 LDIF 是一种标准的文本文件且可以随意的导入导出，所以我们需要有一种“格式”标准化 LFID 文件的写法，这中格式叫做“schema”，schema 用于指定一个目录中锁包含对象的类型，以及每一个类型中的可选属性，我们可以将 schema 理解为面向对象程序设计中的“类”，通过“类”定义出具体的对象，因此其实 LDIF 数据条目则都是通过 schema 数据模型创建出来的具体对象：
**ldapadd 命令用于将 LDIF 文件导入到目录服务数据库中，格式为：“ldapadd [参数] LDIF 文件”。**

| 参数 | 作用 |
|---|---|
| -x | 进行简单认证。 |
| -D | 用于绑定服务器的 dn。 |
| -h: | 目录服务的地址。 |
| -w: | 绑定 dn 的密码。 |
| -f: | 使用 LDIF 文件进行条目添加的文件。 |

添加 cosine 和 nis 模块：

[root@linuxprobe ~]# **cd /etc/openldap/schema/**
[root@linuxprobe schema]# **ldapadd -Y EXTERNAL -H ldapi:/// -D "cn=config" -f cosine.ldif**
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
adding new entry "cn=cosine,cn=schema,cn=config"
[root@linuxprobe schema]# **ldapadd -Y EXTERNAL -H ldapi:/// -D "cn=config" -f nis.ldif**
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
adding new entry "cn=nis,cn=schema,cn=config"

创建/etc/openldap/changes.ldif 文件，并将下面的信息复制进去（注意有一处要修改的地方）：

[root@linuxprobe ~]# **vim /etc/openldap/changes.ldif**
dn: olcDatabase={2}hdb,cn=config
changetype: modify
replace: olcSuffix
olcSuffix: dc=linuxprobe,dc=com

dn: olcDatabase={2}hdb,cn=config
changetype: modify
replace: olcRootDN
olcRootDN: cn=Manager,dc=linuxprobe,dc=com

dn: olcDatabase={2}hdb,cn=config
changetype: modify
replace: olcRootPW
olcRootPW: **此处输入之前生成的密码（如{SSHA}v/GJvGG8SbIuCxhfTDVhkmWEuz2afNIR）**

dn: cn=config
changetype: modify
replace: olcTLSCertificateFile
olcTLSCertificateFile: /etc/openldap/certs/cert.pem

dn: cn=config
changetype: modify
replace: olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/openldap/certs/priv.pem

dn: cn=config
changetype: modify
replace: olcLogLevel
olcLogLevel: -1

dn: olcDatabase={1}monitor,cn=config
changetype: modify
replace: olcAccess
olcAccess: {0}to * by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth" read by dn.base="cn=Manager,dc=linuxprobe,dc=com" read by * none
将新的配置文件更新到 slapd 服务程序：
[root@linuxprobe ~]# **ldapmodify -Y EXTERNAL -H ldapi:/// -f /etc/openldap/changes.ldif**
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
modifying entry "olcDatabase={2}hdb,cn=config"
modifying entry "olcDatabase={2}hdb,cn=config"
modifying entry "olcDatabase={2}hdb,cn=config"
modifying entry "cn=config"
modifying entry "cn=config"
modifying entry "cn=config"
modifying entry "olcDatabase={1}monitor,cn=config"
创建/etc/openldap/base.ldif 文件，并将下面的信息复制进去：
[root@linuxprobe ~]# **vim /etc/openldap/base.ldif**
dn: dc=linuxprobe,dc=com
dc: linuxprobe
objectClass: top
objectClass: domain

dn: ou=People,dc=linuxprobe,dc=com
ou: People
objectClass: top
objectClass: organizationalUnit
dn: ou=Group,dc=linuxprobe,dc=com
ou: Group
objectClass: top
objectClass: organizationalUnit
创建目录的结构服务：
[root@linuxprobe ~]# *ldapadd -x -w linuxprobe -D cn=Manager,dc=linuxprobe,dc=com -f /etc/openldap/base.ldif*
adding new entry "dc=linuxprobe,dc=com"
adding new entry "ou=People,dc=linuxprobe,dc=com"
adding new entry "ou=Group,dc=linuxprobe,dc=com"
创建测试用户并设置其密码：
[root@linuxprobe ~]# useradd -d /home/ldap ldapuser
[root@linuxprobe ~]# passwd ldapuser
Changing password for user ldapuser.
New password: 此处输入要给用户设置的密码
Retype new password: 再次输入密码
passwd: all authentication tokens updated successfully.

设置帐户的迁移（修改第 71 与 74 行）:

```
[root@linuxprobe ~]# vim /usr/share/migrationtools/migrate_common.ph
$DEFAULT_MAIL_DOMAIN = "linuxprobe.com";
$DEFAULT_BASE = "dc=linuxprobe,dc=com";
```

将当前系统中的用户迁移至目录服务:

```
[root@linuxprobe ~]# cd /usr/share/migrationtools/
[root@linuxprobe migrationtools]# grep ":10[0-9][0-9]" /etc/passwd &gt; passwd
[root@linuxprobe migrationtools]# ./migrate_passwd.pl passwd users.ldif
[root@linuxprobe migrationtools]# ldapadd -x -w linuxprobe -D cn=Manager,dc=linuxprobe,dc=com -f users.ldif
adding new entry "uid=linuxprobe,ou=People,dc=linuxprobe,dc=com"
adding new entry "uid=ldapuser,ou=People,dc=linuxprobe,dc=com"
```

将当前系统中的用户组迁移至目录服务:

```
[root@linuxprobe migrationtools]# grep ":10[0-9][0-9]" /etc/group &gt; group
[root@linuxprobe migrationtools]# ./migrate_group.pl group groups.ldif
[root@linuxprobe migrationtools]# ldapadd -x -w linuxprobe -D cn=Manager,dc=linuxprobe,dc=com -f groups.ldif
adding new entry "cn=linuxprobe,ou=Group,dc=linuxprobe,dc=com"
adding new entry "cn=ldapuser,ou=Group,dc=linuxprobe,dc=com"
```

测试 linuxprobe 用户的配置文件:

```
[root@linuxprobe ~]# ldapsearch -x cn=ldapuser -b dc=linuxprobe,dc=com
# extended LDIF
#
# LDAPv3
# base <dc=linuxprobe,dc=com> with scope subtree
# filter: cn=ldapuser
# requesting: ALL
#
# ldapuser, People, linuxprobe.com
dn: uid=ldapuser,ou=People,dc=linuxprobe,dc=com
uid: ldapuser
cn: ldapuser
objectClass: account
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
userPassword:: e2NyeXB0fSQ2JFdtcXFveHFIJFFNaU1pZDAuL01KLnBrR1ZKLkdVSVlWalguTXh
 xLlB5Uk1IeGJseGdkVTBwOUxwcTBJT2huYnkwNFkzdXh1Zi9QaWFpUUtlLk0wUHdQNFpxRXJQV0cv
shadowLastChange: 16713
shadowMin: 0
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: 1001
gidNumber: 1001
homeDirectory: /home/ldapuser
```

```
# ldapuser, Group, linuxprobe.com
dn: cn=ldapuser,ou=Group,dc=linuxprobe,dc=com
objectClass: posixGroup
objectClass: top
cn: ldapuser
userPassword:: e2NyeXB0fXg=
gidNumber: 1001

# search result
search: 2
result: 0 Success

# numResponses: 3
# numEntries: 2
```

安装 httpd 服务程序：

```
[root@linuxprobe ~]# yum install httpd
Loaded plugins: langpacks, product-id, subscription-manager
………………省略部分安装过程………………
Installing:
 httpd              x86_64          2.4.6-17.el7            rhel7          1.2 M
Installing for dependencies:
 apr                x86_64          1.4.8-3.el7             rhel7          103 k
 apr-util           x86_64          1.5.2-6.el7             rhel7          92 k
 httpd-tools        x86_64          2.4.6-17.el7            rhel7          77 k
 mailcap            noarch          2.1.41-2.el7            rhel7          31 k
………………省略部分安装过程………………
Complete!
```

将密钥文件上传至网站目录：

```
[root@linuxprobe ~]# cp /etc/openldap/certs/cert.pem /var/www/html
```

将 httpd 服务程序重启，并添加到开机启动项：

```
[root@linuxprobe ~]# systemctl restart httpd
[root@linuxprobe ~]# systemctl enable httpd
ln -s '/usr/lib/systemd/system/httpd.service' '/etc/systemd/system/multi-user.target.wants/httpd.service'
```

清空防火墙的规则并保存状态：

```
[root@linuxprobe ~]# iptables -F
success
[root@linuxprobe ~]# service iptables save
success
```

在日志记录服务的配置文件中追加下面语句，并重启日志服务：

```
[root@linuxprobe ~]# vim /etc/rsyslog.conf
local4.* /var/log/ldap.log
[root@linuxprobe ~]# systemctl restart rsyslog
```

### 17.2.2 配置 LDAP 客户端

将 LDAP 服务端主机名与 IP 地址的解析记录写入：

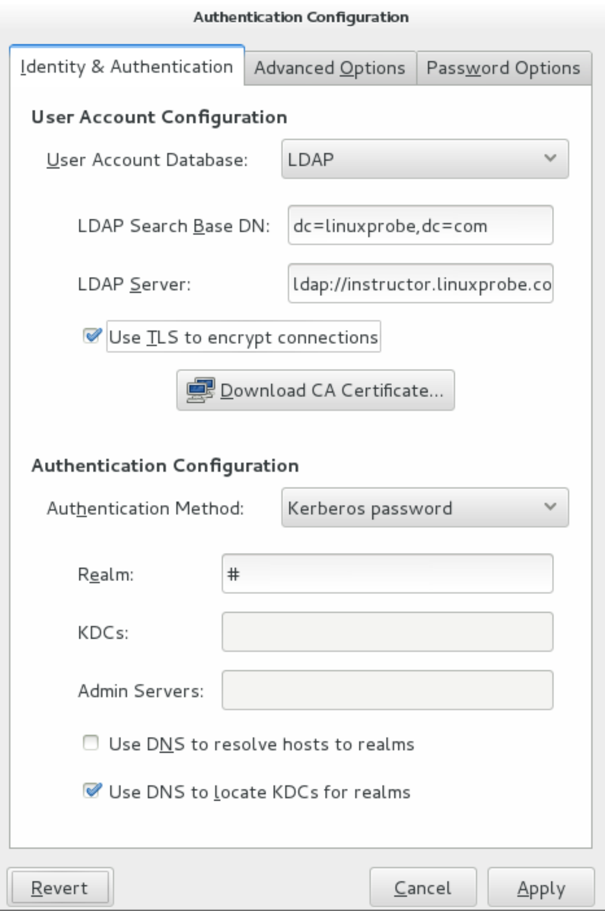
```
[root@linuxprobe ~]# echo "192.168.10.10 instructor.linuxprobe.com" >> /etc/hosts
```

安装相关的软件包：

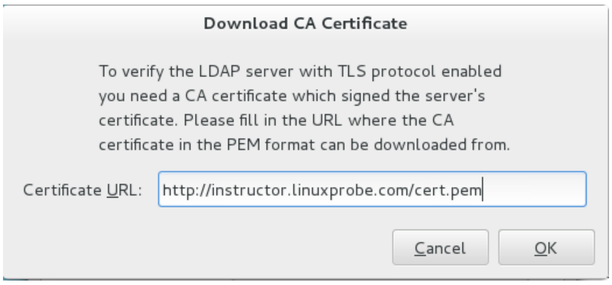
```
[root@linuxprobe Desktop]#   yum install openldap-clients nss-pam-ldapd authconfig-gtk pam_krb5
Loaded plugins: langpacks, product-id, subscription-manager
………………省略部分的安装过程………………
Installing:
 authconfig-gtk          x86_64          6.2.8-8.el7             rhel          105 k
 nss-pam-ldapd           x86_64          0.8.13-8.el7            rhel          159 k
 openldap-clients        x86_64          2.4.39-3.el7            rhel          183 k
 pam_krb5                x86_64          2.4.8-4.el7             rhel          158 k
Installing for dependencies:
 nscd                    x86_64          2.17-55.el7             rhel          250 k
………………省略部分的安装过程………………
Complete!
```

运行系统认证工具，并填写 LDAP 服务信息：

```
[root@linuxprobe ~]# system-config-authentication
```



填写证书地址：




稍等片刻后，验证本地是否已经有了 ldapuser 用户：

[root@linuxprobe ~]# id ldapuser

uid=1001(ldapuser) gid=1001(ldapuser) groups=1001(ldapuser)

此时说明已经可以通过 LDAP 服务端验证了，并且 ldapuser 用户的帐号信息也不会保存在您本地的/etc/passwd 文件中~

**17.3 自动挂载用户目录·**

虽然在客户端已经能够使用 LDAP 验证帐户了，但是当切换到 ldapuser 用户时会提示没有该用户的家目录：

[root@linuxprobe ~]# su - ldapuser

su: warning: cannot change directory to /home/ldapuser: No such file or directory

mkdir: cannot create directory '/home/ldapuser': Permission denied

原因是本机并没有该用户的家目录，我们需要配置 NFS 服务将用户的家目录自动挂载过来：

在 LDAP 服务端添加共享信息（NFS 服务程序已经默认安装，我们之前学过还记得吗？）：

[root@linuxprobe ~]# vim /etc/exports

/home/ldap 192.168.10.20 (rw,sync,root_squash)

重启 nfs-server 服务程序：

[root@linuxprobe ~]# systemctl restart nfs-server

在 LDAP 客户端查看共享信息：

[root@linuxprobe ldap]# showmount -e 192.168.10.10

Export list for 192.168.10.10:

/home/ldap 192.168.10.20

将共享目录挂载到本地：

[root@linuxprobe ~]# mkdir /home/ldap

[root@linuxprobe ldap]# mount -t nfs 192.168.10.10:/home/ldap /home/ldap

再次尝试切换到 ldapuser 用户，这样非常顺利：

[root@linuxprobe ldap]# su - ldapuser

Last login: Tue Oct  6 11:51:25 CST 2015 on pts/3

[ldapuser@linuxprobe ~]$

设置为开机自动挂载：

[root@linuxprobe ~]# vim /etc/fstab

192.168.10.10:/home/ldap /home/ldap nfs defaults 0 0

# 第 18 章 使用 MariaDB 数据库管理系统。

**章节概述:**

MYSQL 数据库管理系统被 Oracle 公司收购后从开源换向到了封闭，导致包括红帽在内的许多 Linux 发行版选择了 MariaDB。本章节将教会您使用 mariaDB 数据库管理工具来管理数据库，学习对数据表单的新建、搜索、更新、插入、删除等常用操作。并且熟练掌握对数据库内用户的创建与授权，数据库的备份与恢复方法，不仅满足了 RHCE 考题要求，还能帮助您的运维工作。

## 18.1 数据库管理系统

我们的生活中无时无刻都在接触到数据，而数据库便是通过指定的组织结构将这数据存储的仓库，并且随着互联网和信息技术的发展，数据库也已经从最初只能存储简单表格发展到了存储海量数据的大型分布式模式。

在信息化社会，充分有效地管理和利用各类信息资源，是进行科学研究和决策管理的前提条件。数据库技术是管理信息系统、办公自动化系统、决策支持系统等各类信息系统的核心部分，是进行科学研究和决策管理的重要技术手段。

数据库管理系统(即 Database Management System)是一种能够对数据库进行建立、使用和维护的软件程序，数据库管理系统通过将计算机中具体的物理数据转换成适合用户理解的抽象逻辑数据，方便用户维护数据库的安全和可用性。

**MYSQL** 是一款大家都非常熟知的数据库管理系统，技术成熟、配置简单、开源免费并且有良好的扩展性，但是 MYSQL 在被 Oracle 公司收购后日渐陷从开源转变为了封闭，缓慢的更新让众多 Linux 发行版（如红帽 RHEL7、Fedora、Centos、OpenSUSE、Slackware 等等）以及诸多已经决定放弃使用这个往日最具人气的数据库管理系统，而转向到了 MariaDB。

**MariaDB** 是 MYSQL 数据库管理系统的一个由开源社区维护的分支产品，完全兼容于 MYSQL，坦白讲虽然 Google 与 Wikipedia 这样的行业巨头已经采用了 MariaDB，但并不意味着会比 MYSQL 有明显的性能提升，而是从技术垄断角度作出的决定。

## 18.2 初始化 mariaDB 服务程序

MariaDB 相对于 MYSQL 来讲确实在功能上有很多扩展特性，比如微秒的支持、线程池、子查询优化、组提交、进度报告等。

安装 mariaDB 服务程序:

```
[root@linuxprobe ~]# yum install mariadb mariadb-server
Loaded plugins: langpacks, product-id, subscription-manager
………………省略部分安装过程………………
Installing:
 mariadb                x86_64        1:5.5.35-3.el7        rhel7        8.9 M
 mariadb-server         x86_64        1:5.5.35-3.el7        rhel7        11 M
Complete!
```

启动 mariadb 服务程序并添加到开机启动项中:

```
[root@linuxprobe ~]# systemctl start mariadb
[root@linuxprobe ~]# systemctl enable mariadb
ln -s '/usr/lib/systemd/system/mariadb.service' '/etc/systemd/system/multi-user.target.wants/mariadb.service'
```

为了保证数据库的安全性，一定要进行初始化工作:

第 1 步：设定 root 用户密码。

第 2 步：删除匿名帐号。
第 3 步：禁止 root 用户从远程登陆。
第 4 步：删除 test 数据库并取消对其的访问权限。
第 5 步：刷新授权表，让初始化后的设定立即生效。
初始化数据库服务程序：
[root@linuxprobe ~]# **mysql_secure_installation**
/usr/bin/mysql_secure_installation: line 379: find_mysql_client: command not found
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!
In order to log into MariaDB to secure it, we'll need the current
password for the root user.  If you've just installed MariaDB, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.
Enter current password for root (enter for none): **当前数据库密码为空，直接敲击回车。**
OK, successfully used password, moving on...
Setting the root password ensures that nobody can log into the MariaDB
root user without the proper authorisation.
Set root password? [Y/n] **y**
New password: **输入要为 root 用户设置的数据库密码。**
Re-enter new password: **重复再输入一次密码。**
Password updated successfully!
Reloading privilege tables..
 ... Success!
By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.
Remove anonymous users? [Y/n] **y（删除匿名帐号）**
 ... Success!
Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.
Disallow root login remotely? [Y/n] **y(禁止 root 用户从远程登陆)**
 ... Success!
By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.
Remove test database and access to it? [Y/n] **y(删除 test 数据库并取消对其的访问权限)**
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!
Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] **y(刷新授权表，让初始化后的设定立即生效)**
 ... Success!
Cleaning up...
All done! If you've completed all of the above steps, your MariaDB
installation should now be secure.
Thanks for using MariaDB!

设置防火墙对数据库服务的允许策略：

[root@linuxprobe ~]# firewall-cmd --permanent --add-service=mysql
success
[root@linuxprobe ~]# firewall-cmd --reload
success

使用 root 用户登陆到数据库中：

[root@linuxprobe ~]# mysql -u root -p
Enter password: 此处输入 root 用户在数据库中的密码。
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 5
Server version: 5.5.35-MariaDB MariaDB Server
Copyright (c) 2000, 2013, Oracle, Monty Program Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]>

查看当前已有的数据库：

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
3 rows in set (0.01 sec)
```

修改当前用户在数据库中的密码（示例中的密码为 redhat）：

MariaDB [(none)]> set password = password('redhat');
Query OK, 0 rows affected (0.00 sec)
MariaDB [(none)]> exit
Bye

使用旧的密码将不能再登陆到数据库：

[root@linuxprobe ~]# mysql -u root -p
Enter password:
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)

### 18.3 管理数据库与表单数据

关系型数据库(DataBase)是由一个或多个数据表单(Table)组成的，数据表单则一般会保存着多个数据记录(Record)。

#### 18.3.1 创建用户并授权

创建一个新的数据库用户：

创建数据库用户的命令:**CREATE USER 用户名@主机名 IDENTIFIED BY ‘密码’;**

MariaDB [(none)]> create user luke@localhost IDENTIFIED BY 'linuxprobe';
Query OK, 0 rows affected (0.00 sec)
进入到 mysql 数据库中：
MariaDB [(none)]> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
查看新创建的**用户、主机、姓名**与**密码**信息：
MariaDB [mysql]> select host,user,password from user where user="luke";
+-----------+------+-------------------------------------------+
| host | user | password |
+-----------+------+-------------------------------------------+
| localhost | luke | *55D9962586BE75F4B7D421E6655973DB07D6869F |
+-----------+------+-------------------------------------------+
1 row in set (0.00 sec)
退出数据库后使用新用户登陆：
MariaDB [mysql]> exit
Bye
[root@linuxprobe ~]# mysql -u luke -p
Enter password: **此处输入 luke 用户的数据库密码**
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 6
Server version: 5.5.35-MariaDB MariaDB Server
Copyright (c) 2000, 2013, Oracle, Monty Program Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
此时只能查看到一个数据库：
MariaDB [(none)]> show databases;
+--------------------+
| Database |
| information_schema |
+--------------------+
1 row in set (0.03 sec)
数据库 GRANT 命令的授权操作常用方案：

| 命令 | 作用 |
| --- | --- |
| GRANT 权限 ON 数据库.表单名称 TO 用户名@主机名 | 对某个特定数据库中的特定表单给予授权。 |
| GRANT 权限 ON 数据库.* TO 用户名@主机名 | 对某个特定数据库中的所有表单给予授权。 |
| GRANT 权限 ON *.* TO 用户名@主机名 | 对所有数据库及所有表单给予授权。 |
| GRANT 权限 1,权限 2 ON 数据库.* TO 用户名@主机名 | 对某个数据库中的所有表单给予多个授权。 |
| GRANT ALL PRIVILEGES ON *.* TO 用户名@主机名 | 对所有数据库及所有表单给予全部授权，(谨慎操作)。 |

切换回 root 用户登陆数据库并进入到 mysql 数据库中：

[root@linuxprobe ~]# mysql -u root -p

MariaDB [(none)]> use mysql;

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

给予 luke 用户对 user 表单的**查询、更新、删除、插入权限**：

MariaDB [mysql]> GRANT SELECT,UPDATE,DELETE,INSERT on mysql.user to luke@localhost;

Query OK, 0 rows affected (0.00 sec)

查看 luke 用户当前的授权：

MariaDB [(none)]> show grants for luke@localhost;

+---------------------------------------------------------------------------------------------------------+

| Grants for luke@localhost |

+---------------------------------------------------------------------------------------------------------+

| GRANT USAGE ON *.* TO 'luke'@'localhost' IDENTIFIED BY PASSWORD '*55D9962586BE75F4B7D421E6655973DB07D6869F' |

| GRANT SELECT, INSERT, UPDATE, DELETE ON `mysql`.`user` TO 'luke'@'localhost' |

+---------------------------------------------------------------------------------------------------------+

2 rows in set (0.00 sec)

再次切换到 luke 用户后查看可用的数据库：

[root@linuxprobe ~]# mysql -u luke -p

MariaDB [(none)]> show databases;

+--------------------+

| Database |

+--------------------+

| information_schema |

| mysql |

+--------------------+

2 rows in set (0.01 sec)

进入到 mysql 数据库中看到 user 表单了：

MariaDB [(none)]> use mysql

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

MariaDB [mysql]> show tables;

+-----------------+

| Tables_in_mysql |

+-----------------+

| user |

+-----------------+

1 row in set (0.01 sec)

切换回 root 用户后取消刚刚全部的授权：

MariaDB [(none)]> revoke SELECT,UPDATE,DELETE,INSERT on mysql.user from luke@localhost;

Query OK, 0 rows affected (0.00 sec)

再次查看 luke 用户的授权：

MariaDB [(none)]> show grants for luke@localhost;

+-------------------------------------------------------------------------------------------------------------+

| Grants for luke@localhost |

+-------------------------------------------------------------------------------------------------------------+

| GRANT USAGE ON *.* TO 'luke'@'localhost' IDENTIFIED BY PASSWORD '*55D9962586BE75F4B7D421E6655973DB07D6869F' |

+-------------------------------------------------------------------------------------------------------------+

1 row in set (0.00 sec)

### 18.3.2 创建数据库与表单

常用的数据库表单管理命令有：

| 用法 | 作用 |
| --- | --- |
| CREATE database 数据库名称。 | 创建新的数据库。 |
| DESCRIBE 表单名称; | 描述表单。 |
| UPDATE 表单名称 SET attribute=新值 WHERE attribute > 原始值; | 更新表单中的数据。 |
| USE 数据库名称; | 指定使用的数据库。 |
| SHOW databases; | 显示当前已有的数据库。 |
| SHOW tables; | 显示当前数据库中的表单。 |
| SELECT * FROM 表单名称; | 从表单中选中某个记录值。 |
| DELETE FROM 表单名 WHERE attribute=值; | 从表单中删除某个记录值。 |

创建一个新的数据库：

MariaDB [(none)]> create database linuxprobe;

Query OK, 1 row affected (0.00 sec)

查看当前已有的数据库：

MariaDB [(none)]> show databases;

+--------------------+

| Database |

+--------------------+

| information_schema |

| **linuxprobe** |

| mysql |

| performance_schema |

+--------------------+

4 rows in set (0.04 sec)

切换到指定的数据库：

MariaDB [(none)]> use linuxprobe;

Database changed

创建新的数据库表单：

```
MariaDB [linuxprobe]> create table mybook (name char(15),price int,pages int);
Query OK, 0 rows affected (0.16 sec)
```

查看表单的结构描述：

```
MariaDB [linuxprobe]> describe mybook;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| name  | char(15) | YES  |     | NULL    |       |
| price | int(11)  | YES  |     | NULL    |       |
| pages | int(11)  | YES  |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
3 rows in set (0.02 sec)
```

### 18.3.3 管理表单数据

向表单内插入新的书籍数据：

```
MariaDB [linuxprobe]> INSERT INTO mybook(name,price,pages) VALUES('linuxprobe','60',518);
Query OK, 1 row affected (0.00 sec)
```

查看表单中的数据值：

```
MariaDB [linuxprobe]> select * from mybook;
+------------+-------+-------+
| name       | price | pages |
+------------+-------+-------+
| linuxprobe |    60 |   518 |
+------------+-------+-------+
1 rows in set (0.01 sec)
```

将价格修改为 55 元：

```
MariaDB [linuxprobe]> update mybook set price=55 ;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

只看书籍的名字和价格：

```
MariaDB [linuxprobe]> select name,price  from mybook;
+------------+-------+
| name       | price |
+------------+-------+
| linuxprobe |    55 |
+------------+-------+
1 row in set (0.00 sec)
```

删除书籍表单中的内容：

```
MariaDB [linuxprobe]> delete from mybook;
Query OK, 1 row affected (0.01 sec)
MariaDB [linuxprobe]> select * from mybook;
Empty set (0.00 sec)
```

连续加入 4 条书籍记录值：

```
MariaDB [linuxprobe]> INSERT INTO mybook(name,price,pages) VALUES('linuxprobe1','30',518);
Query OK, 1 row affected (0.05 sec)
MariaDB [linuxprobe]> INSERT INTO mybook(name,price,pages) VALUES('linuxprobe2','50',518);
Query OK, 1 row affected (0.05 sec)
MariaDB [linuxprobe]> INSERT INTO mybook(name,price,pages) VALUES('linuxprobe3','80',518);
Query OK, 1 row affected (0.01 sec)
MariaDB [linuxprobe]> INSERT INTO mybook(name,price,pages) VALUES('linuxprobe4','100',518);
Query OK, 1 row affected (0.00 sec)
```

where 命令用于在数据库匹配查询的条件，可用的条件有：

| 参数 | 作用 |
|---|---|
| = | 相等。 |
| <>或!= | 不相等。 |
| > | 大于。 |
| < | 小于。 |
| >= | 大于或等于。 |
| <= | 小于或等于。 |
| BETWEEN | 在某个范围内。 |
| LIKE | 搜索一个例子。 |
| IN | 在列中搜索多个值。 |

查看价格大于 75 元的书籍：

```
MariaDB [linuxprobe]> select * from mybook where price>75;
+-------------+-------+-------+
| name        | price | pages |
+-------------+-------+-------+
| linuxprobe3 |    80 |   518 |
| linuxprobe4 |   100 |   518 |
+-------------+-------+-------+
2 rows in set (0.06 sec)
```

搜索价格不等于 80 元的书籍：

```
MariaDB [linuxprobe]> select * from mybook where price!=80;
| name | price | pages |
| linuxprobe1 | 30 | 518 |
| linuxprobe2 | 50 | 518 |
| linuxprobe4 | 100 | 518 |
3 rows in set (0.01 sec)
```

### 18.3.4 数据库的备份与恢复

**mysqldump 命令用于备份数据库数据，格式为："mysqldump [参数] [数据库名称]"。**

| 参数 | 作用 |
| --- | --- |
| -u | 数据库的用户名称。 |
| -p | 密码提示符。 |
| --no-data | 至备份数据库的描述结构，而不要数据。 |
| --lock-all-tables | 备份完成后将不再允许修改数据。 |

将书籍数据库文件（即 linuxprobe）导出到家目录：

[root@linuxprobe ~]# mysqldump -u root -p linuxprobe > /root/linuxprobeDB.dump

Enter password:

删除书籍数据库：

MariaDB [linuxprobe]> drop database linuxprobe;

Query OK, 1 row affected (0.04 sec)

MariaDB [(none)]> show databases;

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
```

3 rows in set (0.02 sec)

创建一个空的数据库：

MariaDB [(none)]> create database linuxprobe;

Query OK, 1 row affected (0.00 sec)

导入刚刚备份的数据库：

[root@linuxprobe ~]# mysql -u root -p linuxprobe < /root/linuxprobeDB.dump

Enter password:

果然又看到了刚刚创建的 mybook 表单：

[root@linuxprobe ~]# mysql -u root -p

MariaDB [(none)]> use linuxprobe;

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

MariaDB [linuxprobe]> show tables;

```
+----------------------+
| Tables_in_linuxprobe |
| mybook               |
+----------------------+
```

1 row in set (0.05 sec)

第 19 章 使用 PXE+Kickstart 部署无人值守安装。

**章节概述：**

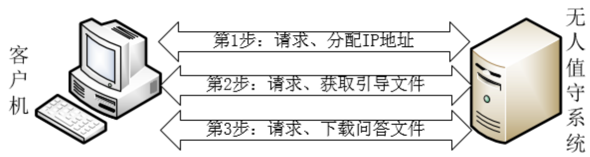
本章节将教会您通过 PXE+DHCP+TFTP+VSftpd+Kickstart 服务程序搭建出无人值守安装系统，从而批量部署客户机系统。这种系统能够实现自动化运维、避免了重复性劳动，帮助提升工作效率，对于运维人员真的是太有帮助了。

## 19.1 无人值守系统

坦白来讲，使用光盘或 U 盘这种传统物理方式安装系统效率真的很低，尤其当需要批量部署系统时更是明显。

一般的机房设备都会在数百台以上，即便购买了数百张系统光盘，那您也必需对每台设备初始化安装向导，免不了会有选错的参数，更何况如此多的设备没有几天肯定装不完吧。

其实我们可以用 **PXE+DHCP+TFTP+VSftpd+Kickstart** 部署出无人值守安装系统，这种系统能够实现自动化运维、避免了重复性劳动，帮助提升工作效率，对于 DHCP 已经是咱们学习过的了，所以这里就不再多说，小伙伴们如果忘记了可以翻去前面章节复习下。




PXE(Preboot execute environment)是一种能够让计算机通过网络启动的引导方式，只要网卡支持 PXE 协议即可使用。

Kickstart 是一种无人值守的安装方式，工作原理就是预先把原本需要运维人员手工填写的参数保存成一个 ks.cfg 文件，当安装过程中出现需要填写参数时则自动匹配 Kickstart 生成的文件，所以只要 Kickstart 文件包含了安装过程中所有需要人工填写的参数，那么运维人员就完全不需要干预，等待安装完毕即可。

简单文本传输协议 TFTP(Trivial File Transfer Protocol)是一种基于 UDP 协议的传输协议，其与前面学习的 vsftpd 服务程序的 FTP 协议有很大不同，TFTP 协议不具备 FTP 的许多功能（例如列出目录，密码认证等等），但 TFTP 协议配置非常简单，而且资源消耗更低，非常适合传输不敏感的文件。

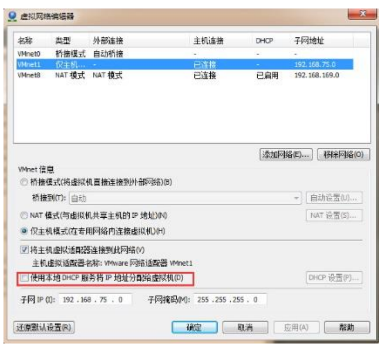
## 19.2 部署相关服务程序

咱们需要依次部署 DHCP、TFTP、SYSLinux、VSFtpd 与 Kickstart 服务，在这之前请先准备两台虚拟机并配置好网卡参数：

| 主机名称 | 操作系统 | IP 地址 |
| --- | --- | --- |
| 无人值守系统 | 红帽 RHEL7 操作系统 | 192.168.10.10 |
| 客户端 | 未安装操作系统 | - |

并确保您的虚拟机软件自带 DHCP 服务功能已经关闭：





### 19.2.1 配置 DHCP 服务程序

安装 dhcpd 服务程序：

```
[root@linuxprobe ~]# yum install dhcp
Loaded plugins: langpacks, product-id, subscription-manager
………………省略部分安装过程………………
Installing:
 dhcp            x86_64          12:4.2.5-27.el7           rhel7          506 k
………………省略部分安装过程………………
Complete!
```

配置 dhcpd 服务程序（将下面的内容复制进去即可，具体参数含义请回顾 dhcpd 服务章节）：

```
[root@linuxprobe ~]# vim /etc/dhcp/dhcpd.conf
allow booting;
allow bootp;
ddns-update-style interim;
ignore client-updates;
subnet 192.168.10.0 netmask 255.255.255.0 {
        option subnet-mask          255.255.255.0;
        option domain-name-servers    192.168.10.10;
        range dynamic-bootp 192.168.10.100 192.168.10.200;
        default-lease-time          21600;
        max-lease-time              43200;
        next-server                 192.168.10.10;
        filename                    "pxelinux.0";
}
```

重启 dhcpd 服务并添加到开机启动项：

```
[root@linuxprobe ~]# systemctl restart dhcpd
[root@linuxprobe ~]# systemctl enable dhcpd
ln -s '/usr/lib/systemd/system/dhcpd.service' '/etc/systemd/system/multi-user.target.wants/dhcpd.service'
```

添加防火墙对 dhcpd 服务允许的规则：

```
[root@linuxprobe ~]# firewall-cmd --permanent --add-service=dhcp
success
[root@linuxprobe ~]# firewall-cmd --reload
success
```

### 19.2.2 配置 TFTP 服务程序

安装 tftp 服务程序：

```
[root@linuxprobe ~]# yum install tftp-server
Loaded plugins: langpacks, product-id, subscription-manager
………………省略部分安装过程………………
Installing:
 tftp-server        x86_64            5.2-11.el7               rhel7            44 k
………………省略部分安装过程………………
Complete!
```

分析 vsftpd 服务程序时发现，FTP 服务器为了随时能够回应客户端的请求必需运行一个长期驻扎在系统中的守护进程，但这样也意味着会有一定的资源浪费，网络守护进程服务程序 xinetd 便是为了解决资源浪费问题而设计的，因为 xinetd 服务程序会同时监听多个系统端口，依据客户端请求的端口再转发给指定的服务程序，而 tftp 便是由 xinetd 服务程序来管理的。

编辑 xinetd 配置文件，启动 TFTP 服务程序：

```
[root@linuxprobe ~.d]# vim /etc/xinetd.d/tftp
//将 disable 的值修改为 no。
service tftp
{
        socket_type             = dgram
        protocol                = udp
        wait                    = yes
        user                    = root
        server                  = /usr/sbin/in.tftpd
        server_args             = -s /var/lib/tftpboot
        disable                 = no
        per_source              = 11
        cps                     = 100 2
        flags                   = IPv4
```

重启 xinetd 服务并添加到开机启动项中：

```
[root@linuxprobe xinetd.d]# systemctl restart xinetd
[root@linuxprobe xinetd.d]# systemctl enable xinetd
```

添加防火墙对 tftp 服务允许的规则：

```
[root@linuxprobe ~]# firewall-cmd --permanent --add-port=69/udp
success
[root@linuxprobe ~]# firewall-cmd --reload
success
```

### 19.2.3 配置 SYSLinux 服务程序

syslinux 是用于提供引导加载的服务程序，目的是简化安装 Linux 系统的时间，安装 syslinux 服务程序：

```
[root@linuxprobe ~]# yum install syslinux
Loaded plugins: langpacks, product-id, subscription-manager
………………省略部分安装过程………………
Installing:
 syslinux            x86_64            4.05-8.el7            rhel7            1.0 M
………………省略部分安装过程………………
Complete!
```

将引导相关文件复制到 tftp 目录以供客户端下载（请确保光盘镜像已挂载到/media/cdrom）：

```
[root@linuxprobe ~]# cd /var/lib/tftpboot
[root@linuxprobe tftpboot]# cp /usr/share/syslinux/pxelinux.0 .
[root@linuxprobe tftpboot]# cp /media/cdrom/images/pxeboot/{vmlinuz,initrd.img} .
[root@linuxprobe tftpboot]# cp /media/cdrom/isolinux/{vesamenu.c32,*.msg} .
```

将引导模板文件复制 tftp 目录：

```
[root@linuxprobe tftpboot]# mkdir pxelinux.cfg
```

[root@linuxprobe tftpboot]# cp /media/cdrom/isolinux/isolinux.cfg pxelinux.cfg/default
编辑引导模板文件：
[root@linuxprobe tftpboot]# *vim pxelinux.cfg/default*
//将第 1 行修改为：
default linux
//将第 64 行修改为：
append initrd=initrd.img inst.stage2=ftp://192.168.10.10 ks=ftp://192.168.10.10/pub/ks.cfg quiet
//将第 70 行修改为：
append initrd=initrd.img inst.stage2=ftp://192.168.10.10 rd.live.check ks=ftp://192.168.10.10/pub/ks.cfg quiet

### 19.2.4 配置 VSFtpd 服务程序

安装 vsftpd 服务程序：
[root@linuxprobe ~]# yum install vsftpd
Loaded plugins: langpacks, product-id, subscription-manager
………………省略部分安装过程………………
Installing:
vsftpd x86_64 3.0.2-9.el7 rhel7 166
k
………………省略部分安装过程………………
Complete!
重启 vsftpd 服务程序并添加到开机启动项：
[root@linuxprobe ~]# *systemctl restart vsftpd*
[root@linuxprobe ~]# *systemctl enable vsftpd*
ln -s '/usr/lib/systemd/system/vsftpd.service' '/etc/systemd/system/multi-user.target.wants/vsftpd.service'
添加防火墙对 vsftpd 服务允许的规则：
[root@linuxprobe ~]# firewall-cmd --permanent --add-service=ftp
success
[root@linuxprobe ~]# firewall-cmd --reload
success
将光盘镜像文件的内容复制到 FTP 目录中（请先确保您的光盘已经挂载到/media/cdrom 目录）：
[root@linuxprobe ~]# cp -r /media/cdrom/* /var/ftp
设置 SELinux 对于 FTP 协议的允许策略：
[root@linuxprobe ~]# setsebool -P ftpd_connect_all_unreserved=on

### 19.2.4 创建 KickStart 应答文件

复制一份应答文件模板并给于权限：
[root@linuxprobe ~]# cp ~/anaconda-ks.cfg /var/ftp/pub/ks.cfg
[root@linuxprobe ~]# chmod +r /var/ftp/pub/ks.cfg
编辑模板文件：
[root@linuxprobe ~]# vim /var/ftp/pub/ks.cfg
//将第 6 行的 cdrom 修改为：
url --url=ftp://192.168.10.10
//将第 21 行的时区修改为：
timezone Asia/Shanghai --isUtc
//将第 28 行修改为：
clearpart --all –initlabel

## 19.3 自动部署客户机

当我们部署好了无人值守系统后，就可以新增一台虚拟机来验证啦：

第 1 步：运行新建虚拟机向导。



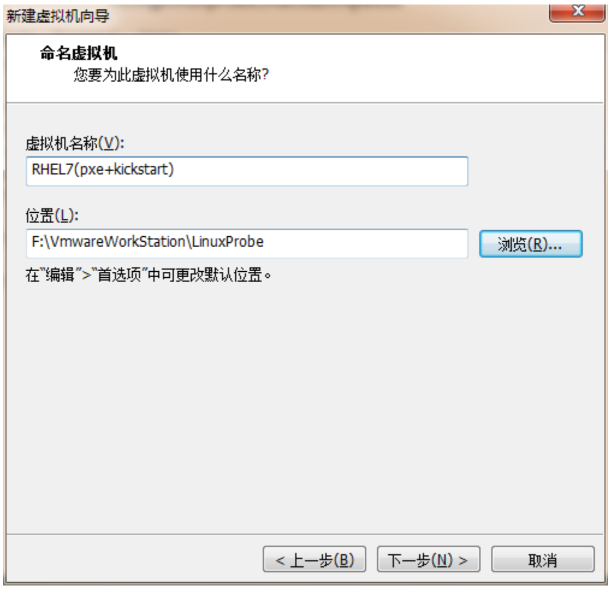

第 2 步：创建一个空白硬盘。



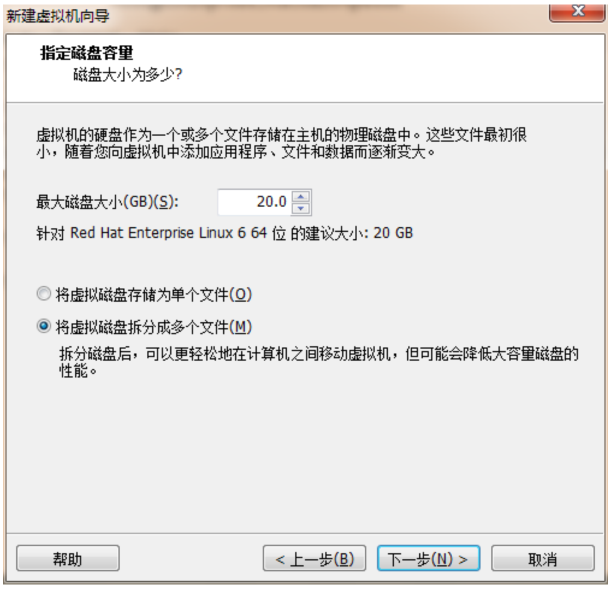

第 3 步：选择虚拟机系统。



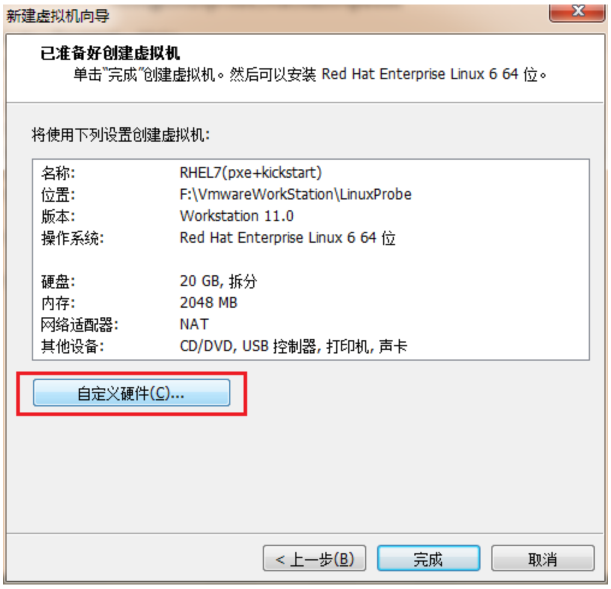

第 4 步：填写系统名称与虚拟机保存路径。



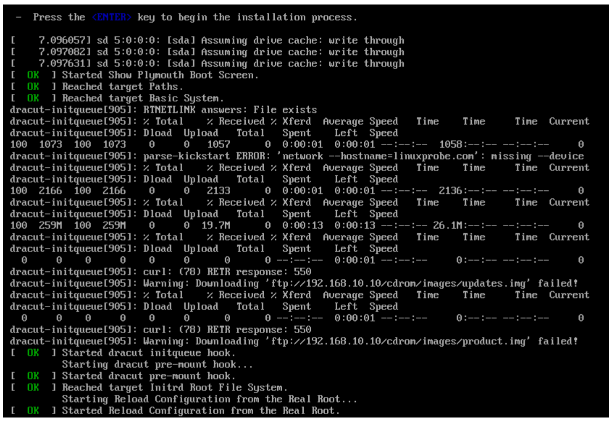
第 5 步：创建硬盘设备（默认大小即可）。



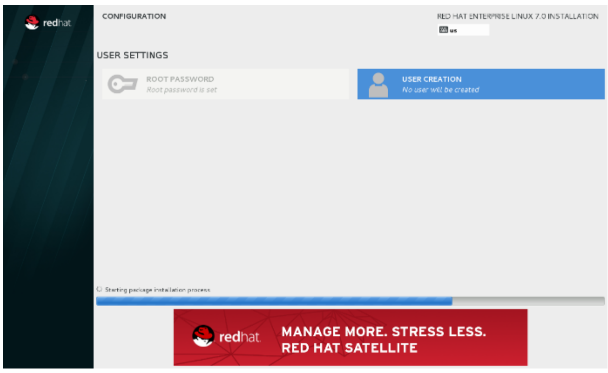

第 6 步：选择自定义硬件，将网卡的类型修改为与无人值守系统一致（此步骤省略）。




当开启客户端虚拟机电源后，会自动化的进行部署系统：




耐心的等待安装中：




大约 20 分钟后系统已经顺利的安装完毕：



# 第 20 章 使用 LNMP 架构部署动态网站环境。

**章节概述：**

本章节将从 Linux 系统的软件安装方式讲起，带领读者分辨 RPM 软件包与源码安装的区别、并能够理解它们的优缺点。

Nginx 是一款相当优秀的用于部署动态网站的服务程序，Nginx 具有不错的稳定性、丰富的功能以及占用较少的系统资源等独特特性。

通过部署 Linux+Nginx+MYSQL+PHP 这四种开源软件，便拥有了一个免费、高效、扩展性强、资源消耗低的 LNMP 动态网站架构了。

## 20.1 源码安装程序

在前面的章节中提到过在红帽软件包管理器（RPM）公布之前要想在 Linux 系统中安装软件只能采取“源码包”的方式安装服务程序，但是源码安装程序真是一件非常困难，耗费耐心的事情，不仅需要运维人员掌握更多的知识、高超的技能、甚至要很有耐心才能安装好一个程序，并且在安装、升级、卸载时还要考虑到其他程序、库的依赖关系，所以我们在前面的课程中都依赖于 YUM 仓库或 RPM 来安装服务程序，但这样也会有一些弊端。



很多软件产品只会以源码包的方式发布，如果只会用 RPM 命令就只能去互联网大海洋中慢慢寻找到由第三方组织或黑客们编写的 RPM 软件包后才能安装程序了，并且源码程序的可移植性非常好，可以针对不同的系统架构而正确运行，但 RPM 软件包则必需严格符合限制使用的平台和架构后才能顺利安装，所以建议即便在工作中可以很舒服的用 Yum 仓库来安装服务程序，源码安装的流程也一定要记清：

**第 1 步，解压文件：**

源码包通常会使用 tar 工具归档然后用 gunzip 或 bzip2 进行压缩，后缀格式会分别为.tar.gz 与 tar.bz2 ，解压方法：

[root@linuxprobe ~]# tar czvf FileName.**tar.gz**

[root@linuxprobe ~]# tar jxvf FileName.**tar.bz2**

**第 2 步，切换到解压后的目录：**

[root@linuxprobe ~]# cd FileDirectory

**第 3 步：准备编译工作：**

在开始安装服务程序之前，需要执行 configure 脚本，他会自动的对当前系统进行一系列的评估，如源文件、软件依赖性库、编译器、汇编器、连接器检查等等，如果有需求，还可以使用 –**prefix** 参数来指定程序的安装路径（很实用），而当脚本检查系统环境符合要求后，则会在当前目录下生成一个 Makefile 文件。

[root@linuxprobe ~]# ./configure –prefix=/usr/local/program

**第 4 步：生成安装程序：**

刚刚生成的 Makefile 文件会保存有系统环境依赖关系和安装规则，接下来需要使用 make 命令来根据 MakeFile 文件提供的规则使用合适的 SHELL 来编译所有依赖的源码，然后 make 命令会生成一个最终可执行的安装程序。

[root@linuxprobe ~]# make

**第 5 步：安装服务程序：**

如果在 configure 脚本阶段中没有使用 –prefix 参数，那么程序一般会被默认安装到/usr/local/bin 目录中。

[root@linuxprobe ~]# make install

**第 6 步：清理临时文件（可选）：**

[root@linuxprobe ~]# make clean

**卸载服务程序的命令（请不要随便执行!!!）：**

[root@linuxprobe ~]# make uninstall

其实读者可能最纳闷的是漫长的 configure 与 make 步骤，RPM 包为什么就可以那么有效率的安装，而不需要检测系统环境呢？

其实原因很简单，RPM 软件包是根据特定系统和平台而指定的，经常一种程序会提供很多 RPM 包的格式（如 i386/x86_64 等等），用户需要找到适合当前自己系统的 RPM 包后才能顺利的安装，而源码包的程序作者肯定希望自己的软件能够被安装到更多的系统中，被更多的用户使用，所以就必需要用 configure 脚本来检查用户当前系统的情况，最终制定出一份可行的安装方案。

## 20.2 部署 LNMP 架构

LNMP（即 Linux+Nginx+MYSQL+PHP）是目前非常热门的动态网站部署架构，一般是指：

Linux:如 RHEL、Centos、Debian、Fedora、Ubuntu 等系统。
Nginx:高性能、低消耗的 HTTP 与反向代理服务程序。
MYSQL:热门常用的数据库管理软件。
PHP:一种能够在服务器端执行的嵌入 HTML 文档的脚本语言。
通过将这四种开源软件部署在一起，便成为了一个免费、高效、扩展性强、资源消耗低的动态网站环境了。

设置防火墙允许数据库与网站服务策略：

```
[root@linuxprobe ~]# iptables -F
[root@linuxprobe ~]# firewall-cmd --permanent --add-service=mysql
success
[root@linuxprobe ~]# firewall-cmd --permanent --add-service=http
success
[root@linuxprobe ~]# firewall-cmd --reload
success
```

下载所有需要使用的软件包到/usr/local/src 目录（17 个文件）：

```
[root@linuxprobe ~]# cd /usr/local/src
[root@linuxprobe src]# wget http://www.linuxprobe.com/Tools/cmake-2.8.11.2.tar.gz
[root@linuxprobe src]# wget http://www.linuxprobe.com/Tools/Discuz_X3.2_SC_GBK.zip
[root@linuxprobe src]# wget http://www.linuxprobe.com/Tools/freetype-2.5.3.tar.gz
[root@linuxprobe src]# wget http://www.linuxprobe.com/Tools/jpegsrc.v9a.tar.gz
[root@linuxprobe src]# wget http://www.linuxprobe.com/Tools/libgd-2.1.0.tar.gz
[root@linuxprobe src]# wget http://www.linuxprobe.com/Tools/libmcrypt-2.5.8.tar.gz
[root@linuxprobe src]# wget http://www.linuxprobe.com/Tools/libpng-1.6.12.tar.gz
[root@linuxprobe src]# wget http://www.linuxprobe.com/Tools/libvpx-v1.3.0.tar.bz2
[root@linuxprobe src]# wget http://www.linuxprobe.com/Tools/mysql-5.6.19.tar.gz
[root@linuxprobe src]# wget http://www.linuxprobe.com/Tools/nginx-1.6.0.tar.gz
[root@linuxprobe src]# wget http://www.linuxprobe.com/Tools/openssl-1.0.1h.tar.gz
[root@linuxprobe src]# wget http://www.linuxprobe.com/Tools/php-5.5.14.tar.gz
[root@linuxprobe src]# wget http://www.linuxprobe.com/Tools/pcre-8.35.tar.gz
[root@linuxprobe src]# wget http://www.linuxprobe.com/Tools/t1lib-5.1.2.tar.gz
[root@linuxprobe src]# wget http://www.linuxprobe.com/Tools/tiff-4.0.3.tar.gz
[root@linuxprobe src]# wget http://www.linuxprobe.com/Tools/yasm-1.2.0.tar.gz
[root@linuxprobe src]# wget http://www.linuxprobe.com/Tools/zlib-1.2.8.tar.gz
```

安装编译工具及库文件（需要安装的程序比较多，请复制全!）:

[root@linuxprobe ~]# **yum install -y apr* autoconf automake bison bzip2 bzip2* compat* cpp curl curl-devel fontconfig fontconfig-devel freetype freetype* freetype-devel gcc gcc-c++ gd gettext gettext-devel glibc kernel kernel-headers keyutils keyutils-libs-devel krb5-devel libcom_err-devel libpng libpng-devel libjpeg* libsepol-devel libselinux-devel libstdc++-devel libtool* libgomp libxml2 libxml2-devel libXpm* libtiff libtiff* make mpfr ncurses* ntp openssl openssl-devel patch pcre-devel perl php-common php-gd policycoreutils telnet t1lib t1lib* nasm nasm* wget zlib-devel**

Loaded plugins: langpacks, product-id, subscription-manager

This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.

………………省略部分安装过程………………

Installing:

………………省略部分安装过程………………

Complete!

安装 cmake 编译工具（解压与编译过程已省略）:

[root@linuxprobe ~]# cd /usr/local/src

[root@linuxprobe src]# ls

```
zlib-1.2.8.tar.gz          libmcrypt-2.5.8.tar.gz   pcre-8.35.tar.gz
cmake-2.8.11.2.tar.gz      libpng-1.6.12.tar.gz     php-5.5.14.tar.gz
Discuz_X3.2_SC_GBK.zip     libvpx-v1.3.0.tar.bz2    t1lib-5.1.2.tar.gz
freetype-2.5.3.tar.gz      mysql-5.6.19.tar.gz      tiff-4.0.3.tar.gz
jpegsrc.v9a.tar.gz         nginx-1.6.0.tar.gz       yasm-1.2.0.tar.gz
libgd-2.1.0.tar.gz         openssl-1.0.1h.tar.gz
```

[root@linuxprobe src]# tar xzvf cmake-2.8.11.2.tar.gz

[root@linuxprobe src]# cd cmake-2.8.11.2/

[root@linuxprobe cmake-2.8.11.2]# ./configure

[root@linuxprobe cmake-2.8.11.2]# make

[root@linuxprobe cmake-2.8.11.2]# make install

### 20.2.1 配置 Mysql 服务

在前面的章节中我们学习了 MariaDB 数据库管理系统，那么这次实验就学习下如何使用 Mysql 来管理数据库吧。

创建用于执行 mysql 服务程序的帐号:

[root@linuxprobe cmake-2.8.11.2]# cd /usr/local/src

[root@linuxprobe src]# useradd mysql -s /sbin/nologin

创建数据库程序和文件的目录，并设置目录的所属与所组:

[root@linuxprobe src]# mkdir -p /usr/local/mysql/var

[root@linuxprobe src]# chown -Rf mysql:mysql /usr/local/mysql/var

安装 Mysql 服务程序（解压与编译过程已省略）:

[root@linuxprobe src]*# tar xzvf mysql-5.6.19.tar.gz*

[root@linuxprobe src]*# cd mysql-5.6.19/*

[root@linuxprobe mysql-5.6.19]*# cmake . -DCMAKE_INSTALL_PREFIX=/usr/local/mysql -DMYSQL_DATADIR=/var/local/mysql/var -DSYSCONFDIR=/etc*

删除系统默认的配置文件:

[root@linuxprobe mysql-5.6.19]# rm -rf /etc/my.cnf

生成系统数据库（生成信息已省略）：
[root@linuxprobe mysql-5.6.19]# *cd /usr/local/mysql/*
[root@linuxprobe mysql]# *./scripts/mysql_install_db --user=mysql --basedir=/usr/local/mysql --datadir=/usr/local/mysql/var*
[root@linuxprobe mysql]# *make*
[root@linuxprobe mysql]# *make install*
创建配置文件的软连接文件：
[root@linuxprobe mysql]# ln -s /usr/local/mysql/my.cnf /etc/my.cnf
将 mysqld 服务程序添加到开机启动项：
[root@linuxprobe mysql]# cp ./support-files/mysql.server /etc/rc.d/init.d/mysqld
[root@linuxprobe mysql]# chmod 755 /etc/init.d/mysqld
[root@linuxprobe mysql]# chkconfig mysqld on
编辑启动项的配置文件：
[root@linuxprobe mysql]# vim /etc/rc.d/init.d/mysqld
//分别修改第 46 与 47 行，basedir 为程序安装路径，datadir 为数据库存放目录。
basedir=/usr/local/mysql
datadir=/usr/local/mysql/var
重启 mysqld 服务程序：
[root@linuxprobe mysql]# systemctl restart mysqld
把 mysql 服务程序命令目录添加到环境变量中（永久生效）：
[root@linuxprobe mysql]# vim /etc/profile
//在配置文件的最下面追加：
export PATH=$PATH:/usr/local/mysql/bin
[root@linuxprobe mysql]# source /etc/profile
将 mysqld 服务程序的库文件链接到默认的位置：
[root@linuxprobe mysql]# mkdir /var/lib/mysql
[root@linuxprobe mysql]# ln -s /usr/local/mysql/lib/mysql /usr/lib/mysql
[root@linuxprobe mysql]# ln -s /usr/local/mysql/include/mysql /usr/include/mysql
[root@linuxprobe mysql]# ln -s /tmp/mysql.sock /var/lib/mysql/mysql.sock
初始化 mysqld 服务程序：
[root@linuxprobe mysql]# **mysql_secure_installation**
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MySQL
SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!
In order to log into MySQL to secure it, we'll need the current
password for the root user. If you've just installed MySQL, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.
Enter current password for root (enter for none):
OK, successfully used password, moving on...
Setting the root password ensures that nobody can log into the MySQL
root user without the proper authorisation.
Set root password? [Y/n] **y**
New password: **输入要为 root 用户设置的数据库密码。**
Re-enter new password: **重复再输入一次密码。**

Password updated successfully!

Reloading privilege tables..

... Success!

By default, a MySQL installation has an anonymous user, allowing anyone

to log into MySQL without having to have a user account created for

them. This is intended only for testing, and to make the installation

go a bit smoother. You should remove them before moving into a

production environment.

Remove anonymous users? [Y/n] **y（删除匿名帐号）**

... Success!

Normally, root should only be allowed to connect from 'localhost'. This

ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] **y(禁止 root 用户从远程登陆)**

... Success!

By default, MySQL comes with a database named 'test' that anyone can

access. This is also intended only for testing, and should be removed

before moving into a production environment.

Remove test database and access to it? [Y/n] **y(删除 test 数据库并取消对其的访问权限)**

- Dropping test database...

... Success!

- Removing privileges on test database...

... Success!

Reloading the privilege tables will ensure that all changes made so far

will take effect immediately.

Reload privilege tables now? [Y/n] **y(刷新授权表，让初始化后的设定立即生效)**

... Success!

All done! If you've completed all of the above steps, your MySQL

installation should now be secure.

Thanks for using MySQL!

Cleaning up...

### 20.2.2 配置 Nginx 服务

Nginx 是一款相当优秀的用于部署动态网站的服务程序，Nginx 最初是为俄罗斯门户站点而设计的网站服务软件，作为作为一款轻量级的网站服务软件，因其稳定性和丰富的功能而深受信赖，但最最最被认可的是**低系统资源、占用内存少且并发能力强**，目前国内如新浪、网易、腾讯等门户站均在使用，市场占有份额一直保持在 15-16%左右。

Nginx 程序的稳定性来自于它采用了分阶段的资源分配技术，使得 CPU 与内存占用率会非常低，所以使用 Nginx 程序部署动态网站环境不仅十分的稳定、高效，而且消耗更少的系统资源，丰富的模块功能也几乎与 Apache 程序数量相同，现在已经完全的支持了 proxy、rewrite、mod_fcgi、ssl、vhosts 等常用模块。而且还支持了热部署技术，即能够可以 7*24 不间断提供服务，即便运行数月也无须重启，而且还可以在不暂停服务的情况下直接对 Nginx 服务程序进行升级。

坦白来讲，虽然 Nginx 程序的代码质量非常高，代码很规范，技术成熟，模块扩展也很容易，但 Nginx 依然存在不少问题，比如 Nginx 是由俄罗斯人创建的，所以在资料文档方面还并不完善，中文教材的质量更是鱼龙混杂，但 Nginx 近年来增长势头迅猛，预测未来应该能够在轻量级 HTTP 服务器市场有不错的未来。

安装 PCRE（Perl 兼容的正则表达式库，解压与编译过程已省略）:

[root@linuxprobe ~]# cd /usr/local/src
[root@linuxprobe src]# mkdir /usr/local/pcre
[root@linuxprobe src]# tar xzvf pcre-8.35.tar.gz
[root@linuxprobe pcre-8.35]# ./configure --prefix=/usr/local/pcre
[root@linuxprobe src]# make
[root@linuxprobe src]# make install

安装 openssl 服务程序（解压与编译过程已省略）:

[root@linuxprobe pcre-8.35]# cd /usr/local/src
[root@linuxprobe src]# mkdir /usr/local/openssl
[root@linuxprobe src]# tar xzvf openssl-1.0.1h.tar.gz
[root@linuxprobe src]# cd pcre-8.35/
[root@linuxprobe pcre-8.35]# ./configure --prefix=/usr/local/openssl
[root@linuxprobe pcre-8.35]# make
[root@linuxprobe pcre-8.35]# make install

把 openssl 服务程序命令目录添加到环境变量中（永久生效）:

[root@linuxprobe pcre-8.35]# vim /etc/profile
//将配置文件最下面的参数追加参数为：
export PATH=$PATH:/usr/local/mysql/bin**:/usr/local/openssl/bin**
**[root@linuxprobe pcre-8.35]# source /etc/profile**

安装 zlib 数据压缩函数库（解压与编译过程已省略）:

[root@linuxprobe pcre-8.35]# cd /usr/local/src
[root@linuxprobe src]# mkdir /usr/local/zlib
[root@linuxprobe src]# tar xzvf zlib-1.2.8.tar.gz
[root@linuxprobe zlib-1.2.8]# ./configure --prefix=/usr/local/zlib
[root@linuxprobe zlib-1.2.8]# make
[root@linuxprobe zlib-1.2.8]# make install

创建用于执行 nginx 服务的用户:

[root@linuxprobe zlib-1.2.8]# cd /usr/local/src
[root@linuxprobe src]# useradd www -s /sbin/nologin

安装 nginx 服务程序（openssl,zlib,pcre 要写成源码解压路径!!!）:

[root@linuxprobe src]# tar xzvf nginx-1.6.0.tar.gz
[root@linuxprobe src]# cd nginx-1.6.0/
[root@linuxprobe nginx-1.6.0]# ./configure --prefix=/usr/local/nginx --without-http_memcached_module --user=www --group=www --with-http_stub_status_module --with-http_ssl_module --with-http_gzip_static_module --with-openssl=/usr/local/src/openssl-1.0.1h --with-zlib=/usr/local/src/zlib-1.2.8 --with-pcre=/usr/local/src/pcre-8.35
[root@linuxprobe nginx-1.6.0]# make
[root@linuxprobe nginx-1.6.0]# make install

创建 nginx 程序脚本（将下面的参数直接复制进去即可）：

[root@linuxprobe nginx-1.6.0]# **vim /etc/rc.d/init.d/nginx**

```
#!/bin/bash
# nginx - this script starts and stops the nginx daemon
# chkconfig: - 85 15
# description: Nginx is an HTTP(S) server, HTTP(S) reverse \
# proxy and IMAP/POP3 proxy server
# processname: nginx
# config: /etc/nginx/nginx.conf
# config: /usr/local/nginx/conf/nginx.conf
# pidfile: /usr/local/nginx/logs/nginx.pid
# Source function library.
. /etc/rc.d/init.d/functions
# Source networking configuration.
. /etc/sysconfig/network
# Check that networking is up.
[ "$NETWORKING" = "no" ] && exit 0
nginx="/usr/local/nginx/sbin/nginx"
prog=$(basename $nginx)
NGINX_CONF_FILE="/usr/local/nginx/conf/nginx.conf"
[ -f /etc/sysconfig/nginx ] && . /etc/sysconfig/nginx
lockfile=/var/lock/subsys/nginx
make_dirs() {
# make required directories
user=‘$nginx -V 2>&1 | grep "configure arguments:" | sed ’s/[^*]*--user=\([^ ]*\).*/\1/g’ -‘
	if [ -z "‘grep $user /etc/passwd‘" ]; then
		useradd -M -s /bin/nologin $user
	fi
options=‘$nginx -V 2>&1 | grep ’configure arguments:‘‘
for opt in $options; do
	if [ ‘echo $opt | grep ’.*-temp-path‘‘ ]; then
		value=‘echo $opt | cut -d "=" -f 2‘
		if [ ! -d "$value" ]; then
			# echo "creating" $value
			mkdir -p $value && chown -R $user $value
		fi
	fi
done
}
start() {
[ -x $nginx ] || exit 5
[ -f $NGINX_CONF_FILE ] || exit 6
make_dirs
echo -n $"Starting $prog: "
```

```
daemon $nginx -c $NGINX_CONF_FILE
retval=$?
echo
[ $retval -eq 0 ] && touch $lockfile
return $retval
}
stop() {
echo -n $"Stopping $prog: "
killproc $prog -QUIT
retval=$?
echo
[ $retval -eq 0 ] && rm -f $lockfile
return $retval
}
restart() {
#configtest || return $?
stop
sleep 1
start
}
reload() {
#configtest || return $?
echo -n $"Reloading $prog: "
killproc $nginx -HUP
RETVAL=$?
echo
}
force_reload() {
restart
}
configtest() {
$nginx -t -c $NGINX_CONF_FILE
}
rh_status() {
status $prog
}
rh_status_q() {
rh_status >/dev/null 2>&1
}
case "$1" in
start)
        rh_status_q && exit 0
        $1
        ;;
```

```
stop)
        rh_status_q || exit 0
        $1
        ;;
restart|configtest)
$1
;;
reload)
        rh_status_q || exit 7
        $1
        ;;
force-reload)
        force_reload
        ;;
status)
        rh_status
        ;;
condrestart|try-restart)
        rh_status_q || exit 0
        ;;
*)
echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart|reload|force-reload|configtest}"
exit 2
esac
[root@linuxprobe nginx-1.6.0]# chmod 755 /etc/rc.d/init.d/nginx
```

重启 nginx 服务程序并添加到开机启动项：

```
[root@linuxprobe nginx-1.6.0]# /etc/rc.d/init.d/nginx restart
Restarting nginx (via systemctl):                    [  OK  ]
[root@linuxprobe nginx-1.6.0]# chkconfig nginx on
```

此时可以通过访问 IP 来判断 nginx 服务是否顺利运行：




### 20.2.3 配置 php 服务

安装 yasm 汇编器（解压与编译过程已省略）：

```
[root@linuxprobe nginx-1.6.0]# cd /usr/local/src
[root@linuxprobe src]# tar zxvf yasm-1.2.0.tar.gz
[root@linuxprobe src]# cd yasm-1.2.0
[root@linuxprobe yasm-1.2.0]# ./configure
```

[root@linuxprobe yasm-1.2.0]# make
[root@linuxprobe yasm-1.2.0]# make install
安装 libmcrypt 加密算法扩展库（解压与编译过程已省略）:
[root@linuxprobe yasm-1.2.0]# cd /usr/local/src
[root@linuxprobe src]# tar zxvf libmcrypt-2.5.8.tar.gz
[root@linuxprobe src]# cd libmcrypt-2.5.8
[root@linuxprobe libmcrypt-2.5.8]# ./configure
[root@linuxprobe libmcrypt-2.5.8]# make
[root@linuxprobe libmcrypt-2.5.8]# make install
安装 libvpx 视频编码器（解压与编译过程已省略）:
[root@linuxprobe libmcrypt-2.5.8]# cd /usr/local/src
[root@linuxprobe src]# tar xjvf libvpx-v1.3.0.tar.bz2
[root@linuxprobe src]# cd libvpx-v1.3.0
[root@linuxprobe libvpx-v1.3.0]# ./configure --prefix=/usr/local/libvpx --enable-shared --enable-vp9
[root@linuxprobe libvpx-v1.3.0]# make
[root@linuxprobe libvpx-v1.3.0]# make install
安装 Tiff 标签图像文件格式（解压与编译过程已省略）:
[root@linuxprobe libvpx-v1.3.0]# cd /usr/local/src
[root@linuxprobe src]# tar zxvf tiff-4.0.3.tar.gz
[root@linuxprobe src]# cd tiff-4.0.3
[root@linuxprobe tiff-4.0.3]# ./configure --prefix=/usr/local/tiff --enable-shared
[root@linuxprobe tiff-4.0.3]# make
[root@linuxprobe tiff-4.0.3]# make install
安装 libpng 图片（png 格式）函数库（解压与编译过程已省略）:
[root@linuxprobe tiff-4.0.3]# cd /usr/local/src
[root@linuxprobe src]# tar zxvf libpng-1.6.12.tar.gz
[root@linuxprobe src]# cd libpng-1.6.12
[root@linuxprobe libpng-1.6.12]# ./configure --prefix=/usr/local/libpng --enable-shared
[root@linuxprobe libpng-1.6.12]# make
[root@linuxprobe libpng-1.6.12]# make install
安装 freetype 字体引擎（解压与编译过程已省略）:
[root@linuxprobe libpng-1.6.12]# cd /usr/local/src
[root@linuxprobe src]# tar zxvf freetype-2.5.3.tar.gz
[root@linuxprobe src]# cd freetype-2.5.3
[root@linuxprobe freetype-2.5.3]# ./configure --prefix=/usr/local/freetype --enable-shared
[root@linuxprobe freetype-2.5.3]# make
[root@linuxprobe freetype-2.5.3]# make install
安装 libpng 图片（jpeg 格式）函数库（解压与编译过程已省略）:
[root@linuxprobe freetype-2.5.3]# cd /usr/local/src
[root@linuxprobe src]# tar zxvf jpegsrc.v9a.tar.gz
[root@linuxprobe src]# cd jpeg-9a
[root@linuxprobe jpeg-9a]# ./configure --prefix=/usr/local/jpeg --enable-shared
[root@linuxprobe jpeg-9a]# make
[root@linuxprobe jpeg-9a]# make install

安装 libgd 图像处理程序（解压与编译过程已省略）:

[root@linuxprobe jpeg-9a]# cd /usr/local/src

[root@linuxprobe src]# tar zxvf libgd-2.1.0.tar.gz

[root@linuxprobe src]# cd libgd-2.1.0

[root@linuxprobe libgd-2.1.0]# ./configure --prefix=/usr/local/libgd --enable-shared --with-jpeg=/usr/local/jpeg --with-png=/usr/local/libpng --with-freetype=/usr/local/freetype --with-fontconfig=/usr/local/freetype --with-xpm=/usr/ --with-tiff=/usr/local/tiff --with-vpx=/usr/local/libvpx

[root@linuxprobe libgd-2.1.0]# make

[root@linuxprobe libgd-2.1.0]# make install

安装 t1lib 图片生成函数库（解压与编译过程已省略）:

[root@linuxprobe cd libgd-2.1.0]# cd /usr/local/src

[root@linuxprobe src]# tar zxvf t1lib-5.1.2.tar.gz

[root@linuxprobe src]# cd t1lib-5.1.2

[root@linuxprobe t1lib-5.1.2]# ./configure --prefix=/usr/local/t1lib --enable-shared

[root@linuxprobe t1lib-5.1.2]# make

[root@linuxprobe t1lib-5.1.2]# make install

将函数库文件放至合适的位置:

[root@linuxprobe t1lib-5.1.2]# cd /usr/local/src

[root@linuxprobe src]# ln -s /usr/lib64/libltdl.so /usr/lib/libltdl.so

[root@linuxprobe src]# cp -frp /usr/lib64/libXpm.so* /usr/lib/

安装 php 服务程序（命令比较长，请一定要复制完整!!!）:

[root@linuxprobe src]# tar -zvxf php-5.5.14.tar.gz

[root@linuxprobe src]# cd php-5.5.14

[root@linuxprobe php-5.5.14]# export LD_LIBRARY_PATH=/usr/local/libgd/lib

[root@linuxprobe php-5.5.14]# ./configure --prefix=/usr/local/php --with-config-file-path=/usr/local/php/etc --with-mysql=/usr/local/mysql --with-mysqli=/usr/local/mysql/bin/mysql_config --with-mysql-sock=/tmp/mysql.sock --with-pdo-mysql=/usr/local/mysql --with-gd --with-png-dir=/usr/local/libpng --with-jpeg-dir=/usr/local/jpeg --with-freetype-dir=/usr/local/freetype --with-xpm-dir=/usr/ --with-vpx-dir=/usr/local/libvpx/ --with-zlib-dir=/usr/local/zlib --with-t1lib=/usr/local/t1lib --with-iconv --enable-libxml --enable-xml --enable-bcmath --enable-shmop --enable-sysvsem --enable-inline-optimization --enable-opcache --enable-mbregex --enable-fpm --enable-mbstring --enable-ftp --enable-gd-native-ttf --with-openssl --enable-pcntl --enable-sockets --with-xmlrpc --enable-zip --enable-soap --without-pear --with-gettext --enable-session --with-mcrypt --with-curl --enable-ctype

[root@linuxprobe php-5.5.14]# make

[root@linuxprobe php-5.5.14]# make install

复制 php 服务程序的配置文件到安装目录:

[root@linuxprobe php-5.5.14]# cp php.ini-production /usr/local/php/etc/php.ini

删除默认的 php 配置文件:

[root@linuxprobe php-5.5.14]# rm -rf /etc/php.ini

创建 php 配置文件的软连接到/etc/目录中:

[root@linuxprobe php-5.5.14]# *cp /usr/local/php/etc/php-fpm.conf.default /usr/local/php/etc/php-fpm.conf*

[root@linuxprobe php-5.5.14]# *ln -s /usr/local/php/etc/php-fpm.conf /etc/php-fpm.conf*

[root@linuxprobe php-5.5.14]# *ln -s /usr/local/php/etc/php.ini /etc/php.ini*

编辑 php 服务程序的配置文件:

[root@linuxprobe php-5.5.14]# vim /usr/local/php/etc/php-fpm.conf

//将第 25 行参数前面的分号去掉。

pid = run/php-fpm.pid

//修改第 148 和 149 行，将 user 与 group 修改为 www。

user = www

group = www

添加 php-fpm 服务程序到开机启动项：

[root@linuxprobe php-5.5.14]# cp sapi/fpm/init.d.php-fpm /etc/rc.d/init.d/php-fpm

[root@linuxprobe php-5.5.14]# chmod +x /etc/rc.d/init.d/php-fpm

[root@linuxprobe php-5.5.14]# chkconfig php-fpm on

为了保障网站的安全性，禁用掉不安全的功能：

[root@linuxprobe php-5.5.14]# vim /usr/local/php/etc/php.ini

//修改第 305 行的 disable_functions 参数，追加参数为：

disable_functions = passthru,exec,system,chroot,scandir,chgrp,chown,shell_exec,proc_open,proc_get_status,ini_alter,ini_alter,ini_restore,dl,openlog,syslog,readlink,symlink,popepassthru,stream_socket_server,escapeshellcmd,dll,popen,disk_free_space,checkdnsrr,checkdnsrr,getservbyname,getservbyport,disk_total_space,posix_ctermid,posix_get_last_error,posix_getcwd,posix_getegid,posix_geteuid,posix_getgid,posix_getgrgid,posix_getgrnam,posix_getgroups,posix_getlogin,posix_getpgid,posix_getpgrp,posix_getpid,posix_getppid,posix_getpwnam,posix_getpwuid,posix_getrlimit,posix_getsid,posix_getuid,posix_isatty,posix_kill,posix_mkfifo,posix_setegid,posix_seteuid,posix_setgid,posix_setpgid,posix_setsid,posix_setuid,posix_strerror,posix_times,posix_ttyname,posix_uname

配置 nginx 服务程序支持 php：

[root@linuxprobe php-5.5.14]# vim /usr/local/nginx/conf/nginx.conf

//将第 2 行前面的#号去掉并修改为 user www www ;

//将第 44 行参数修改为 index index.html index.htm index.php;

//将第 64-71 行前面的#号去掉，修改为：

```
location ~ \.php$ {
   root              html;
   fastcgi_pass      127.0.0.1:9000;
   fastcgi_index     index.php;
   fastcgi_param     SCRIPT_FILENAME $document_root$fastcgi_script_name;
   include           fastcgi_params;
   }
```

重启 nginx 与 php-fpm 服务程序：

[root@linuxprobe php-5.5.14]# systemctl restart nginx

[root@linuxprobe php-5.5.14]# systemctl restart php-fpm

### 20.3 搭建 discuz 论坛

将 discuz 论坛数据放至网站目录(解压过程已省略)：

[root@linuxprobe ~ ]# cd /usr/local/src/

[root@linuxprobe src]# unzip Discuz_X3.2_SC_GBK.zip

[root@linuxprobe src]# rm -rf /usr/local/nginx/html/{index.html,50x.html}

[root@linuxprobe src]# mv upload/* /usr/local/nginx/html/

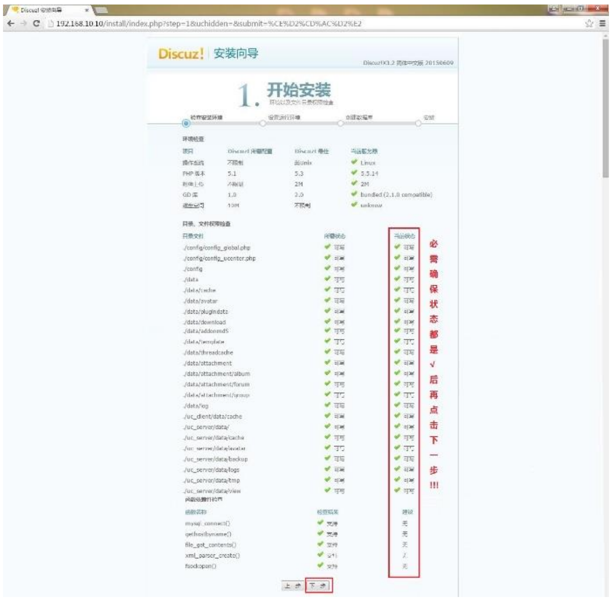
[root@linuxprobe src]# chown -Rf www:www /usr/local/nginx/html

[root@linuxprobe src]# chmod -Rf 755 /usr/local/nginx/html

第 1 步，接受许可协议：




第 2 步，检查部署环境：




第 3 步，选择全新安装 discuzX 论坛：




第 4 步，填写数据库与论坛管理员信息：



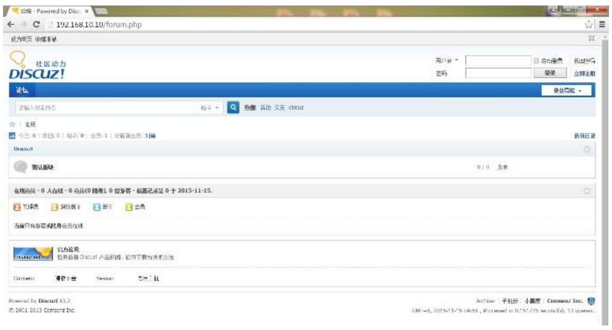
第 5 步，等待安装完毕：




第 6 步，discuz 论坛顺利安装完毕：




第 7 步，访问论坛主页面：

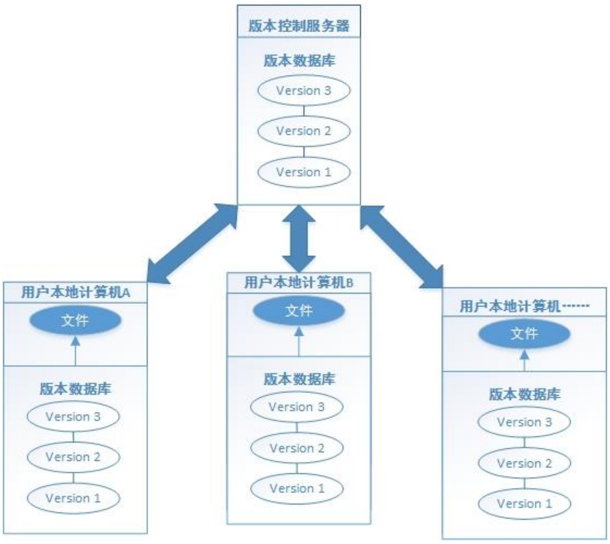



本章结束，您可以在此写下笔记：

第 21 章 使用 Git 分布式版本控制系统。

### 21.1 分布式版本控制系统

我想大家还记得前面章节中谈到过 Linus torvalds 在 1991 年时发布了 Linux 操作系统吧，从那以后 Linux 系统变不断发展壮大，因为 Linux 系统开源的特性，所以一直接受着来自全球 Linux 技术爱好者的贡献，志愿者们通过邮件向 Linus 发送着自己编写的源代码文件，然后由 Linus 本人通过手工的方式将代码合并，但这样不仅没有效率，而且真的是太痛苦了。
一直到 2002 年，Linux 系统经过十余年的不断发展，代码库已经庞大到无法再让 Linus 通过手工的方式管理了，但是 Linus 真的很不喜欢 CVS 或者 Subversion 版本控制系统，于是商业公司 BitMover 决定将其公司的 BitKeeper 分布式版本控制系统授权给 Linux 开发社区来免费使用，当时的 BitKeeper 可以比较文件内容的不同，还能够将出错的文档还原到历史某个状态，Linus 终于放下了心里的石头。




**分布式版本控制流程图，原稿。**

CVS 和 Subversion 属于传统的版本控制系统，而分布式版本控制系统最大的特点是不需要每次提交都把文件推送到版本控制服务器，而是采用分布式版本库的机制，使得每个开发人员都够从服务器中克隆一份完整的版本库到自己计算机本地，不必再完全依赖于版本控制服务器，使得源代码的发布和合并更加方便，并且因为数据都在自己本地，不仅效率提高了，而且即便我们离开了网络依然可以执行提交文件、查看历史版本记录、创建分支等等操作，真的是开发者的福音啊。

就这样平静的度过了三年时间，但是 Linux 社区聚集着太多的黑客人物，2005 年时，那位曾经开发 Samba 服务程序的 Andrew 因为试图破解 BitKeeper 软件协议而激怒了 BitMover 公司，当即决定不再向 Linux 社区提供免费的软件授权了，此时的 Linus 其实也早已有自己编写分布式版本控制系统的打算了，于是便用 C 语言创建了 Git 分布式版本控制系统，并上传了 Linux 系统的源代码。

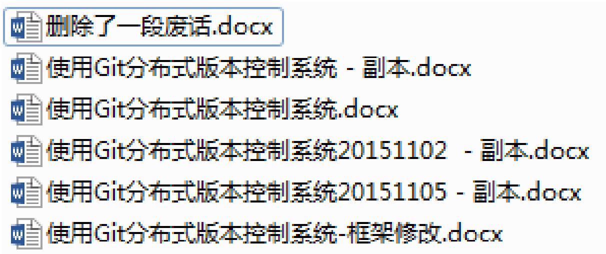
Git 不仅是一款开源的分布式版本控制系统，而且有其独特的功能特性，例如大多数的分布式版本控制系统只会记录每次文件的变化，说白了就是只会关心文件的内容变化差异，而 Git 则是关注于文件数据整体的变化，直接会将文件提交时的数据保存成快照，而非仅记录差异内容，并且使用 SHA-1 加密算法保证数据的完整性。

Git 为了提高效率，对于没有被修改的文件，则不会重复存储，而是创建一个链接指向之前存储过的文件。




Git 提交流程图，原稿。

其实从发明计算机至今，编写文档工作早已融入到每个人的生活之中，但为了完成一篇好文章，一定免不了反复的修改，而人们的思维如此活跃，一不小心就变成了这个样子：




无意中就创建出了这么多乱七八糟的文档，但是那个才是我想要的版本呢？而且又担心要删除的文档中可能保留有某个不错的想法，删除后就不能找回了。更要命的是，有些章节还需要团队一起编写，于是需要把文件传输给他们，等到编写后再传回来，最后由我逐条对照差别后将新的内容添加进去，这样真的是太麻烦了，我们更希望看到这样的记录吧：

编辑

| 版本 | 用户 | 说明 | 日期 |
|---|---|---|---|
| 1 | Ronny | 创建 Git 章节文档 | 10/12 13:48 |
| 2 | Dave | 新增 Git 命令介绍 | 10/15 12:19 |
| 3 | Aaron | 新增 Github 使用方法 | 10/20 8:32 |
| 4 | Kim | 改正文章中的错别字 | 10/30 15:17 |

### 21.2 使用 Git 服务程序

在正式使用前，我们还需要弄清楚 Git 的三种重要模式，分别是已提交、已修改和已暂存：

已提交(committed):表示数据文件已经顺利提交到 Git 数据库中。

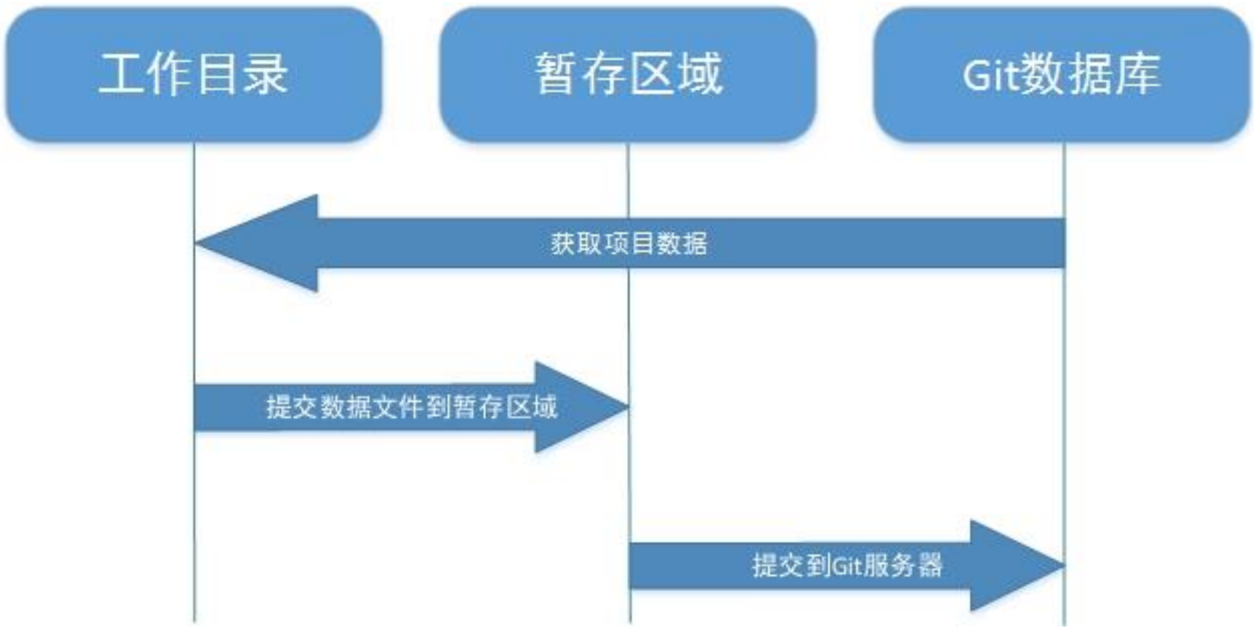
已修改(modified):表示数据文件已经被修改，但未被保存到 Git 数据库中。

已暂存(staged):表示数据文件已经被修改，并会在下次提交时提交到 Git 数据库中。
提交前的数据文件可能会被随意修改或丢失，但只要把文件快照顺利提交到 Git 数据库中，那就可以完全放心了，流程为：

1.在工作目录中修改数据文件。

2.将文件的快照放入暂存区域。

3.将暂存区域的文件快照提交到 Git 仓库中。




Git 的工作流程图，原稿。

执行 yum 命令来安装 Git 服务程序：

```
[root@linuxprobe ~]# yum install -y git
Loaded plugins: langpacks, product-id, subscription-manager
………………省略部分安装过程………………
Installing:
 git                    x86_64        1.8.3.1-4.el7           rhel7        4.3 M
Installing for dependencies:
 perl-Error             noarch        1:0.17020-2.el7         rhel7         32 k
```

```
 perl-Git                    noarch          1.8.3.1-4.el7              rhel7           52  k
 perl-TermReadKey           x86_64          2.30-20.el7                  rhel7           31  k
………………省略部分安装过程………………
Complete!
```

首次安装 Git 服务程序后需要设置下用户名称、邮件信息和编辑器，这些信息会随着文件每次都提交到 Git 数据库中，用于记录提交者的信息，而 Git 服务程序的配置文档通常会有三份，针对当前用户和指定仓库的配置文件优先级最高：

| 配置文件 | 作用 |
| --- | --- |
| /etc/gitconfig | 保存着系统中每个用户及仓库通用配置信息。 |
| ~/.gitconfig<br>~/.config/git/config | 针对于当前用户的配置信息。 |
| 工作目录/.git/config | 针对于当前仓库数据的配置信息。 |

第一个要配置的是你个人的用户名称和电子邮件地址，这两条配置很重要，每次 Git 提交时都会引用这两条信息，记录是谁提交了文件，并且会随更新内容一起被永久纳入历史记录：

```
[root@linuxprobe ~]# git config --global user.name "Liu Chuan"
[root@linuxprobe ~]# git config --global user.email "root@linuxprobe.com"
```

设置 vim 为默认的文本编辑器：

```
[root@linuxprobe ~]# git config --global core.editor vim
```

嗯，此时查看下刚刚配置的 Git 工作环境信息吧：

```
[root@linuxprobe ~]# git config --list
user.name=Liu Chuan
user.email=root@linuxprobe.com
core.editor=vim
```

### 21.2.1 提交数据

我们可以简单的把工作目录理解成是一个被 Git 服务程序管理的目录，Git 会时刻的追踪目录内文件的改动，另外在安装好了 Git 服务程序后，默认就会创建好了一个叫做 master 的分支，我们直接可以提交数据到了。

创建本地的工作目录：

```
[root@linuxprobe ~]# mkdir linuxprobe
[root@linuxprobe ~]# cd linuxprobe/
```

将该目录转初始化成 Git 的工作目录：

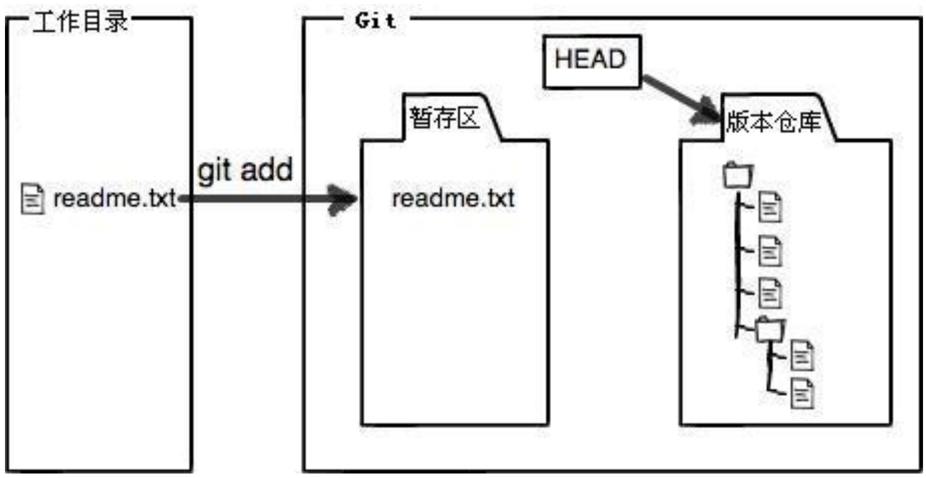
```
[root@linuxprobe linuxprobe]# git init
Initialized empty Git repository in /root/linuxprobe/.git/
```

Git 只能追踪类似于 txt 文件、网页、程序源码等文本文件的内容变化，而不能判断图片、视频、可执行命令等这些二进制文件的内容变化，所以先来尝试往里面写入一个新文件吧。

```
[root@linuxprobe linuxprobe]# echo "Initialization Git repository" > readme.txt
```

**将该文件添加到暂存区：**

```
[root@linuxprobe linuxprobe]# git add readme.txt
```

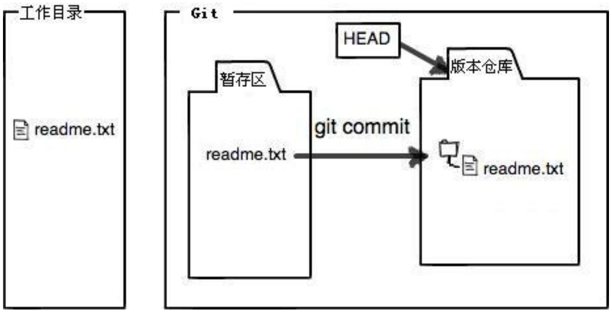



添加到暂存区后再次修改文件的内容：

```
[root@linuxprobe linuxprobe]# echo "Something not important" >> readme.txt
```

将暂存区的文件提交到 Git 版本仓库，命令格式为“git commit -m “提交说明””：

```
[root@linuxprobe linuxprobe]# git commit -m "add the readme file"
[master (root-commit) f091d8b] add the readme file
 1 file changed, 1 insertion(+)
 create mode 100644 readme.txt
```




查看当前工作目录的状态（咦，为什么文件还是提示被修改了？）：

```
[root@linuxprobe linuxprobe]# git status
# On branch master
# Changes not staged for commit:
#   (use "git add ..." to update what will be committed)
#   (use "git checkout -- ..." to discard changes in working directory)
#
#         modified:   readme.txt
#
no changes added to commit (use "git add" and/or "git commit -a")
```

因为提交操作只是将文件在暂存区中的快照版本提交到 Git 版本数据库，所以当你将文件添加到暂存区后，如果又对文件做了修改，请一定要再将文件添加到暂存区后提交到 Git 版本数据库：

第一次修改 -> git add -> 第二次修改 -> git add -> git commit
查看当前文件内容与 Git 版本数据库中的差别：

```
[root@linuxprobe linuxprobe]# git diff readme.txt
diff --git a/readme.txt b/readme.txt
index cb06697..33d16d0 100644
--- a/readme.txt
+++ b/readme.txt
@@ -1 +1,2 @@
 Initialization Git repository
+Something not important
```

那么现在把文件提交到 Git 版本数据库吧：

```
[root@linuxprobe linuxprobe]# git add readme.txt
[root@linuxprobe linuxprobe]# git commit -m "added a line of words"
[master 346f1ab] add a line of words
 1 file changed, 1 insertion(+)
```

再来查看下当前 Git 版本仓库的状态：

```
[root@linuxprobe linuxprobe]# git status
# On branch master
nothing to commit, working directory clean
```

有些时候工作目录内的文件会比较多，懒的把文件一个个提交到暂存区，可以先设置下要忽略上传的文件（写入到”工作目录/.gitignore“文件中），然后使用”git add .”命令来将当前工作目录内的所有文件都一起添加到暂存区域。

```
//忽略所有以.a 为后缀的文件。
*.a
//但是 lib.a 这个文件除外，依然会被提交。
!lib.a
//忽略 build 目录内的所有文件。
build/
//忽略 build 目录内以 txt 为后缀的文件。
build/*.txt
//指定忽略名字为 git.c 的文件。
git.c
```

先在工作目录中创建一个名字为 git.c 的文件：

```
[root@linuxprobe linuxprobe]# touch git.c
```

然后创建忽略文件列表：

```
[root@linuxprobe linuxprobe]# vim .gitignore
git.c
```

添加将当前工作目录中的所有文件快照上传到暂存区：

```
[root@linuxprobe linuxprobe]# git add .
[root@linuxprobe linuxprobe]# git commit -m "add the .gitignore file"
[master c8dfed8] add the .gitignore file
 1 file changed, 1 insertion(+)
 create mode 100644 .gitignore
```

经过刚刚的实验，大家一定发现“添加到暂存区”真是个很麻烦的步骤，虽然使用暂存区的方式可以让提交文件更加的准确，但有时却略显繁琐，如果对要提交的文件完全有把握，我们完全可以追加-a 参数，这样 Git 会将以前所有追踪过的文件添加到暂存区后自动的提交，从而跳过了上传暂存区的步骤，再来修改下文件：

```
[root@linuxprobe linuxprobe]# echo "Modified again" >> readme.txt
```

文件被直接提交到 Git 数据库：

```
[root@linuxprobe linuxprobe]# git commit -a -m "Modified again"
[master 711d5cf] Modified again
 1 file changed, 1 insertion(+)
[root@linuxprobe linuxprobe]# git status
# On branch master
nothing to commit, working directory clean
```

比如想把 git.c 也提交上去，便可以这样强制添加文件：

```
[root@linuxprobe linuxprobe]# git add -f git.c
```

然后重新提交一次（即修改上次的提交操作）：

```
[root@linuxprobe linuxprobe]# git commit --amend
Modified again
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# Changes to be committed:
#   (use "git reset HEAD^1 ..." to unstage)
#
#       new file:   git.c
```

```
#       modified:   readme.txt
#
//我们简单浏览下提交描述，然后输入:wq!保存并退出。
[master 7d8c026] Modified again
 2 files changed, 1 insertion(+)
 create mode 100644 git.c
```

### 21.2.2 移除数据

有些时候会向把已经添加到暂存区的文件移除，但仍然希望文件在工作目录中不丢失，换句话说，就是把文件从追踪清单中删除。
先添加一个新文件，并上传到暂存区：

```
[root@linuxprobe linuxprobe]# touch database
[root@linuxprobe linuxprobe]# git add database
```

查看当前的 Git 状态：

```
[root@linuxprobe linuxprobe]# git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD ..." to unstage)
#
#         new file:   database
#
```

将该文件从 Git 暂存区域的追踪列表中移除（并不会删除当前工作目录内的数据文件）：

```
[root@linuxprobe linuxprobe]# git rm --cached database
rm 'database'
[root@linuxprobe linuxprobe]# ls
database  git.c  readme.txt
```

此时文件已经是未追踪状态了：

```
[root@linuxprobe linuxprobe]# git status
```

```
# On branch master
# Untracked files:
#    (use "git add ..." to include in what will be committed)
#
#             database
nothing added to commit but untracked files present (use "git add" to track)
```

而如果我们向将文件数据从 Git 暂存区和工作目录中一起删除，可以这样操作：
再将 database 文件提交到 Git 暂存区：

```
[root@linuxprobe linuxprobe]# git add .
```

使用 git rm 命令可以直接删除暂存区内的追踪信息及工作目录内的数据文件：
但如果在删除之前数据文件已经被放入到暂存区域的话，Git 会担心你勿删未提交的文件而提示报错信息，此时可追加强制删除-f 参数。

```
[root@linuxprobe linuxprobe]# git rm -f database
rm 'database'
[root@linuxprobe linuxprobe]# ls
git.c   readme.txt
```

查看当前 Git 的状态：

```
[root@linuxprobe linuxprobe]# git status
# On branch master
nothing to commit, working directory clean
```

### 21.2.3 移动数据

Git 不像其他版本控制系统那样跟踪文件的移动操作，如果要修改文件名称，则需要使用 git mv 命令：

```
[root@linuxprobe linuxprobe]# git mv readme.txt introduction.txt
```

发现下次提交时会有一个改名操作：

```
[root@linuxprobe linuxprobe]# git status
```

```
# On branch master
# Changes to be committed:
#   (use "git reset HEAD ..." to unstage)
#
#       renamed:    readme.txt -> introduction.txt
#
```

提交文件到 Git 版本仓库：

```
[root@linuxprobe linuxprobe]# git commit -m "changed name"
[master 76a7f52] changed name
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename readme.txt => introduction.txt (100%)
```

其实我们还可以这样来修改文件名，首先将工作目录下的数据文件改名：

```
[root@linuxprobe linuxprobe]# mv introduction.txt readme.txt
```

然后删除 Git 版本仓库内的文件快照：

```
[root@linuxprobe linuxprobe]# git rm introduction.txt
rm 'introduction.txt'
```

最后再将新的文件添加进入：

```
[root@linuxprobe linuxprobe]# git add readme.txt
[root@linuxprobe linuxprobe]# git commit -m "changed the file name again"
[master 026055f] changed the file name again
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename introduction.txt => readme.txt (100%)
```

### 21.2.4 历史记录

在完成上面的实验后，我们已经不知不觉有了很多次的提交操作了，可以用 git log 命令来查看提交历史记录：

```
[root@linuxprobe linuxprobe]# git log
```

```
commit 026055f9a45bc83babc7efa54820914321bc0ea1
Author: Liu Chuan <root@linuxprobe.com>
Date: Tue Dec 15 17:58:13 2015 +0800
 changed the file name again
commit 76a7f52538fa9bde09b646603d90dd948cfe80c4
Author: Liu Chuan <root@linuxprobe.com>
Date: Tue Dec 15 17:53:11 2015 +0800
 changed name
commit 7d8c026df1ce5eb27494c30104180ae3a82c80b6
Author: Liu Chuan <root@linuxprobe.com>
Date: Tue Dec 15 17:28:23 2015 +0800
 Modified again
commit c8dfed8cac2174c482a1af601657a08b7dcc8950
Author: Liu Chuan <root@linuxprobe.com>
Date: Tue Dec 15 17:25:55 2015 +0800
 add the .gitignore file
commit 346f1ab36ec532f649fbd573cd23a3229684b969
Author: Liu Chuan <root@linuxprobe.com>
Date: Tue Dec 15 17:23:37 2015 +0800
 add a line of words
commit f091d8be1a94c8fecd6abc2fa31730fe2a7a8887
Author: Liu Chuan <root@linuxprobe.com>
Date: Tue Dec 15 17:22:07 2015 +0800
 add the readme file
```

像上面直接执行 git log 命令后会看到所有的更新记录（按时间排序，最近更新的会在上面），历史记录会除了保存文件快照，还会详细的记录着文件 SHA-1 校验和，作者的姓名，邮箱及更新时间，如果只想看最近几条记录，可以直接这样操作：

```
[root@linuxprobe linuxprobe]# git log -2
commit 026055f9a45bc83babc7efa54820914321bc0ea1
Author: Liu Chuan <root@linuxprobe.com>
```

```
Date: Tue Dec 15 17:58:13 2015 +0800
 changed the file name again
commit 76a7f52538fa9bde09b646603d90dd948cfe80c4
Author: Liu Chuan <root@linuxprobe.com>
Date: Tue Dec 15 17:53:11 2015 +0800
 changed name
```

我也常用-p 参数来展开显示每次提交的内容差异，例如仅查看最近一次的差异：

```
[root@linuxprobe linuxprobe]# git log -p -1
commit 026055f9a45bc83babc7efa54820914321bc0ea1
Author: Liu Chuan <root@linuxprobe.com>
Date: Tue Dec 15 17:58:13 2015 +0800
 changed the file name again
diff --git a/introduction.txt b/introduction.txt
deleted file mode 100644
index f3c8232..0000000
--- a/introduction.txt
+++ /dev/null
@@ -1,3 +0,0 @@
-Initialization Git repository
-Something not important
-Modified again
diff --git a/readme.txt b/readme.txt
new file mode 100644
index 0000000..f3c8232
--- /dev/null
+++ b/readme.txt
@@ -0,0 +1,3 @@
+Initialization Git repository
+Something not important
```

```
+Modified again
```

我们还可以使用 --stat 参数来简要的显示数据增改行数，这样就能够看到提交中修改过的内容、对文件添加或移除的行数，并在最后列出所有增减行的概要信息（仅看最近两次的提交历史）：

```
[root@linuxprobe linuxprobe]# git log --stat -2
commit 026055f9a45bc83babc7efa54820914321bc0ea1
Author: Liu Chuan <root@linuxprobe.com>
Date: Tue Dec 15 17:58:13 2015 +0800
 changed the file name again
 introduction.txt | 3 ---
 readme.txt | 3 +++
 2 files changed, 3 insertions(+), 3 deletions(-)
commit 76a7f52538fa9bde09b646603d90dd948cfe80c4
Author: Liu Chuan <root@linuxprobe.com>
Date: Tue Dec 15 17:53:11 2015 +0800
 changed name
 introduction.txt | 3 +++
 readme.txt | 3 ---
 2 files changed, 3 insertions(+), 3 deletions(-)
```

还有一个超级常用的 --pretty 参数，它可以根据不同的格式为我们展示提交的历史信息，比如每行显示一条提交记录：

```
[root@linuxprobe linuxprobe]# git log --pretty=oneline
026055f9a45bc83babc7efa54820914321bc0ea1 changed the file name again
76a7f52538fa9bde09b646603d90dd948cfe80c4 changed name
7d8c026df1ce5eb27494c30104180ae3a82c80b6 Modified again
c8dfed8cac2174c482a1af601657a08b7dcc8950 add the .gitignore file
346f1ab36ec532f649fbd573cd23a3229684b969 add a line of words
f091d8be1a94c8fecd6abc2fa31730fe2a7a8887 add the readme file
```

以更详细的模式输出提交的历史记录：

```
[root@linuxprobe linuxprobe]# git log --pretty=fuller -2
```

```
commit 026055f9a45bc83babc7efa54820914321bc0ea1
Author: Liu Chuan <root@linuxprobe.com>
AuthorDate: Tue Dec 15 17:58:13 2015 +0800
Commit: Liu Chuan <root@linuxprobe.com>
CommitDate: Tue Dec 15 17:58:13 2015 +0800

 changed the file name again

commit 76a7f52538fa9bde09b646603d90dd948cfe80c4
Author: Liu Chuan <root@linuxprobe.com>
AuthorDate: Tue Dec 15 17:53:11 2015 +0800
Commit: Liu Chuan <root@linuxprobe.com>
CommitDate: Tue Dec 15 17:53:11 2015 +0800

 changed name
```

还可以使用 format 参数来指定具体的输出格式，这样非常便于后期编程的提取分析哦，常用的格式有：

| | |
|---|---|
| %s | 提交说明。 |
| %cd | 提交日期。 |
| %an | 作者的名字。 |
| %cn | 提交者的姓名。 |
| %ce | 提交者的电子邮件。 |
| %H | 提交对象的完整 SHA-1 哈希字串。 |
| %h | 提交对象的简短 SHA-1 哈希字串。 |
| %T | 树对象的完整 SHA-1 哈希字串。 |
| %t | 树对象的简短 SHA-1 哈希字串。 |
| %P | 父对象的完整 SHA-1 哈希字串。 |
| %p | 父对象的简短 SHA-1 哈希字串。 |

| %ad | 作者的修订时间。 |
|---|---|

另外作者和提交者是不同的，作者才是对文件作出实际修改的人，而提交者只是最后将此文件提交到 Git 版本数据库的人。

查看当前所有提交记录的简短 SHA-1 哈希字串与提交者的姓名：

```
[root@linuxprobe linuxprobe]# git log --pretty=format:"%h %cn"
026055f Liu Chuan
76a7f52 Liu Chuan
7d8c026 Liu Chuan
c8dfed8 Liu Chuan
346f1ab Liu Chuan
f091d8b Liu Chuan
```

### 21.2.5 还原数据

还原数据是每一个版本控制的基本功能，先来随意修改下文件吧：

```
[root@linuxprobe linuxprobe]# echo "Git is a version control system" >> readme.txt
```

然后将文件提交到 Git 版本数据库：

```
[root@linuxprobe linuxprobe]# git add readme.txt
[root@linuxprobe linuxprobe]# git commit -m "Introduction software"
[master 5cee15b] Introduction software
 1 file changed, 1 insertion(+)
```

此时觉得写的不妥，向要还原某一次提交的文件快照：

```
[root@linuxprobe linuxprobe]# git log --pretty=oneline
5cee15b32d78259985bac4e0cbb0cdad72ab68ad Introduction software
026055f9a45bc83babc7efa54820914321bc0ea1 changed the file name again
76a7f52538fa9bde09b646603d90dd948cfe80c4 changed name
7d8c026df1ce5eb27494c30104180ae3a82c80b6 Modified again
c8dfed8cac2174c482a1af601657a08b7dcc8950 add the .gitignore file
```

```
346f1ab36ec532f649fbd573cd23a3229684b969 add a line of words
f091d8be1a94c8fecd6abc2fa31730fe2a7a8887 add the readme file
```

Git 服务程序中有一个叫做 HEAD 的版本指针，当用户申请还原数据时，其实就是将 HEAD 指针指向到某个特定的提交版本而已，但是因为 Git 是分布式版本控制系统，所以不可能像 SVN 那样使用 1、2、3、4 来定义每个历史的提交版本号，为了避免历史记录冲突，故使用了 SHA-1 计算出十六进制的哈希字串来区分每个提交版本，像刚刚最上面最新的提交版本号就是 5cee15b32d78259985bac4e0cbb0cdad72ab68ad，另外默认的 HEAD 版本指针会指向到最近的一次提交版本记录哦，而上一个提交版本会叫 HEAD^，上上一个版本则会叫做 HEAD^^，当然一般会用 HEAD~5 来表示往上数第五个提交版本哦~。

好啦，既然我们已经锁定了要还原的历史提交版本，就可以使用 get reset 命令来还原数据了：

```
[root@linuxprobe linuxprobe]# git reset --hard HEAD^
HEAD is now at 026055f changed the file name again
```

再来看下文件的内容吧（怎么样，内容果然已经还原了吧~）：

```
[root@linuxprobe linuxprobe]# cat readme.txt
Initialization Git repository
Something not important
Modified again
```

刚刚的操作实际上就是改变了一下 HEAD 版本指针的位置，说白了就是你将 HEAD 指针放在那里，那么你的当前工作版本就会定位在那里，要想把内容再还原到最新提交的版本，先看查看下提交版本号吧：

```
[root@linuxprobe linuxprobe]# git log --pretty=oneline
026055f9a45bc83babc7efa54820914321bc0ea1 changed the file name again
76a7f52538fa9bde09b646603d90dd948cfe80c4 changed name
7d8c026df1ce5eb27494c30104180ae3a82c80b6 Modified again
c8dfed8cac2174c482a1af601657a08b7dcc8950 add the .gitignore file
346f1ab36ec532f649fbd573cd23a3229684b969 add a line of words
f091d8be1a94c8fecd6abc2fa31730fe2a7a8887 add the readme file
```

怎么搞得？竟然没有了 Introduction software 这个提交版本记录？？
原因很简单，因为我们当前的工作版本是历史的一个提交点，这个历史提交点还没有发生过 Introduction software 更新记录，所以当然就看不到了，要是想“还原到未来”的历史更新点，可以用 git reflog 命令来查看所有的历史记录：

```
[root@linuxprobe linuxprobe]# git reflog
026055f HEAD@{0}: reset: moving to HEAD^
```

```
5cee15b HEAD@{1}: commit: Introduction software
026055f HEAD@{2}: commit: changed the file name again
76a7f52 HEAD@{3}: commit: changed name
7d8c026 HEAD@{4}: commit (amend): Modified again
711d5cf HEAD@{5}: commit: Modified again
c8dfed8 HEAD@{6}: commit: add the .gitignore file
346f1ab HEAD@{7}: commit: add a line of words
f091d8b HEAD@{8}: commit (initial): add the readme file
```

找到历史还原点的 SHA-1 值后，就可以还原文件了，另外 SHA-1 值没有必要写全，Git 会自动去匹配：x

```
[root@linuxprobe linuxprobe]# git reset --hard 5cee15b
HEAD is now at 5cee15b Introduction software
```

如是只是想把某个文件内容还原，就不必这么麻烦，直接用 git checkout 命令就可以的，先随便写入一段话：

```
[root@linuxprobe linuxprobe]# echo "Some mistakes words" >> readme.txt
[root@linuxprobe linuxprobe]# cat readme.txt
Initialization Git repository
Something not important
Modified again
Git is a version control system
Some mistakes words
```

哎呀，我们突然发现不应该写一句话的，可以手工删除（当内容比较多的时候会很麻烦），还可以将文件内容从暂存区中恢复：

```
[root@linuxprobe linuxprobe]# git checkout -- readme.txt
[root@linuxprobe linuxprobe]# cat readme.txt
Initialization Git repository
Something not important
Modified again
Git is a version control system
```

这其中是有一套规则哦，如果暂存区中有该文件，则直接从暂存区恢复，如果暂存区没有该文件，则将还原成最近一次文件提交时的快照。

### 21.2.6 管理标签

当版本仓库内的数据有个大的改善或者功能更新，我们经常会打一个类似于软件版本号的标签，这样通过标签就可以将版本库中的某个历史版本给记录下来，方便我们随时将特定历史时期的数据取出来用，另外打标签其实只是像某个历史版本做了一个指针，所以一般都是瞬间完成的，感觉很方便吧。
在 Git 中打标签非常简单，给最近一次提交的记录打个标签：

```
[root@linuxprobe linuxprobe]# git tag v1.0
```

查看所有的已有标签：

```
[root@linuxprobe linuxprobe]# git tag
v1.0
```

查看此标签的详细信息：

```
[root@linuxprobe linuxprobe]# git show v1.0
commit d316fb2970d9cc476c1fcfc6b4950bfd6ebcb795
Author: Liu Chuan
Date:   Tue Dec 15 19:59:08 2015 +0800
    Introduction software
diff --git a/readme.txt b/readme.txt
index f3c8232..955efba 100644
--- a/readme.txt
+++ b/readme.txt
@@ -1,3 +1,4 @@
 Initialization Git repository
 Something not important
 Modified again
+Git is a version control system
```

还可以创建带有说明的标签，用-a 指定标签名，-m 指定说明文字：

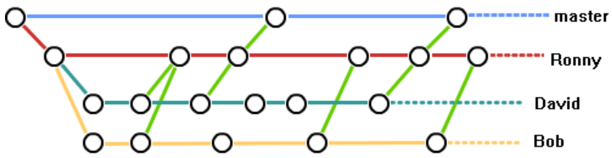
```
[root@linuxprobe linuxprobe]# git tag v1.2 -m "version 1.2 released" d316fb
```

我们为同一个提交版本设置了两次标签，来把之前的标签删除吧：

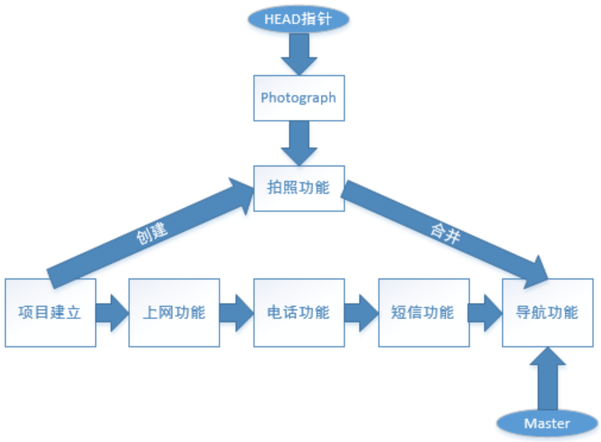
```
[root@linuxprobe linuxprobe]# git tag -d v1.0
Deleted tag 'v1.0' (was d316fb2)
```

### 21.3 管理分支结构

分支即是平行空间，假设你在为某个手机系统研发拍照功能，代码已经完成了 80%，但如果将这不完整的代码直接提交到 git 仓库中，又有可能影响到其他人的工作，此时我们便可以在该软件的项目之上创建一个名叫“拍照功能”的分支，这种分支只会属于你自己，而其他人看不到，等代码编写完成后再与原来的项目主分支合并下即可，这样即能保证代码不丢失，又不影响其他人的工作。




一般在实际的项目开发中，我们要尽量保证 master 分支是非常稳定的，仅用于发布新版本，平时不要随便直接修改里面的数据文件，而工作的时候则可以新建不同的工作分支，等到工作完成后在合并到 master 分支上面，所以团队的合作分支看起来会像上面图那样。




另外如前面所讲，git 会将每次的提交操作串成一个时间线，而在前面的实验中实际都是在对 master 分支进行操作，Git 会在创建分支后默认创建一个叫做 Photograph 的指针，所以我们还需要再将 HEAD 指针切换到“Photograph”的位置才正式使用上了新分支哦，这么说起来可能比较抽象，赶紧学习下面的实验吧。

#### 21.3.1 创建分支

首先创建分支：

```
[root@linuxprobe linuxprobe]# git branch linuxprobe
```

切换至分支：

```
[root@linuxprobe linuxprobe]# git checkout linuxprobe
```

查看当前分支的情况（会列出该仓库中所有的分支，当前的分支前有＊号）：

```
[root@linuxprobe linuxprobe]# git branch
```

我们对文件追加再一行字符串吧：

```
[root@linuxprobe linuxprobe]# echo "Creating a new branch is quick." >> readme.txt
```

将文件提交到 git 仓库：

```
[root@linuxprobe linuxprobe]# git add readme.txt
[root@linuxprobe linuxprobe]# git commit -m "new branch"
```

为了让大家更好理解分支的作用，我们在提交文件后再切换回 master 分支：

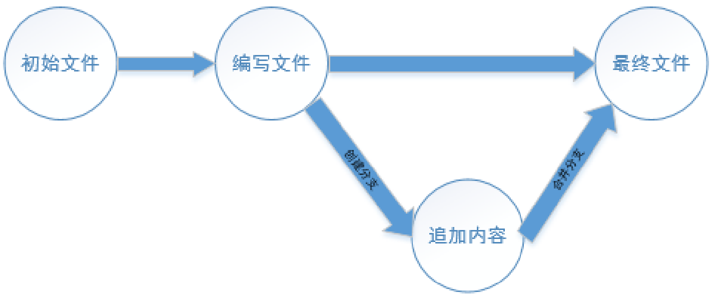
```
[root@linuxprobe linuxprobe]# git checkout master
```

然后查看下文件内容，发现并没有新追加的字符串哦：

```
[root@linuxprobe linuxprobe]# cat readme.txt
```

### 21.3.2 合并分支

现在，我们想把 linuxprobe 的工作成果合并到 master 分支上了，则可以使用”git merge”命令来将指定的的分支与当前分支合并：




```
[root@linuxprobe linuxprobe]# git merge linuxprobe
```

查看合并后的 readme.txt 文件：

```
[root@linuxprobe linuxprobe]# cat readme.txt
```

确认合并完成后，就可以放心地删除 linuxprobe 分支了：

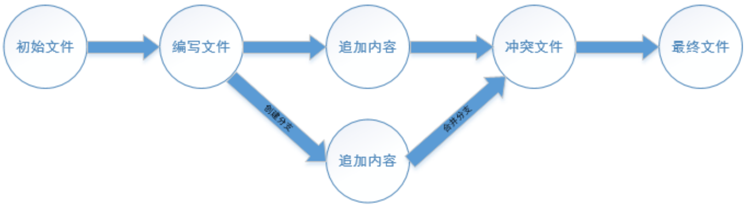
```
[root@linuxprobe linuxprobe]# git branch -d linuxprobe
```

删除后，查看 branch，就只剩下 master 分支了：

```
[root@linuxprobe linuxprobe]# git branch
* master
```

### 21.3.3 分支冲突

但是 Git 并不能每次都为我们自动的合并分支，当遇到了内容冲突比较复杂的情况，则必须手工将差异内容处理点，比如这样的情况：




创建分支并切换到该分支命令：git checkout -b 分支名称
创建一个新分支并切换到该分支命令：

```
[root@linuxprobe linuxprobe]# git checkout -b linuxprobe
```

修改 readme.txt 文件内容：

```
[root@linuxprobe linuxprobe]# vim readme.txt
Creating a new branch is quick & simple.
```

在 linuxprobe 分支上提交：

```
[root@linuxprobe linuxprobe]# git add readme.txt
[root@linuxprobe linuxprobe]# git commit -m "Creating a new branch is quick & simple. "
[feature1 75a857c] AND simple 1 file changed, 1 insertion(+), 1 deletion(-)
```

切换到 master 分支：

```
[root@linuxprobe linuxprobe]# git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
```

在在 master 分支上修改 readme.txt 文件同一行的内容：

```
[root@linuxprobe linuxprobe]# vim readme.txt
```

```
Creating a new branch is quick AND simple.
```

提交至 Git 版本仓库：

```
[root@linuxprobe linuxprobe]# git add readme.txt
Creating a new branch is quick AND simple.
[root@linuxprobe linuxprobe]# git commit -m "Creating a new branch is quick AND simple."
[master 400b400] & simple 1 file changed, 1 insertion(+), 1 deletion(-)
```

那么此时，我们在 master 与 linuxprobe 分支上都分别对中 readme.txt 文件进行了修改并提交了，那这种情况下 Git 就没法再为我们自动的快速合并了，它只能告诉我们 readme.txt 文件的内容有冲突，需要手工处理冲突的内容后才能继续合并：

```
[root@linuxprobe linuxprobe]# git merge linuxprobe
Auto-merging readme.txt CONFLICT (content): Merge conflict in readme.txt Automatic merge failed; fix conf
licts and then commit the result.
```

冲突的内容为：

```
[root@localhost linuxprobe]# vim readme.txt
Git is a distributed version control system.
Git is free software distributed under the GPL.
Git has a mutable index called stage.
Git tracks changes of files.
<<<<<<<HEAD
Creating a new branch is quick & simple.
=======
Creating a new branch is quick AND simple.
>>>>>>> linuxprobe
```

Git 用< <<<<<<，=======，>>>>>>>分割开了各个分支冲突的内容，我们需要手工的删除这些符号，并将内容修改为：

Creating a new branch is quick and simple. 解决冲突内容后则可顺利的提交：

```
[root@linuxprobe linuxprobe]# git add readme.txt
```

```
[root@linuxprobe linuxprobe]# git commit -m "conflict fixed"
[master 59bc1cb] conflict fixed
```

查看 Git 历史提交记录(可以看到分支的变化)：

```
[root@linuxprobe linuxprobe]# git log --graph --pretty=oneline --abbrev-commit
*   59bc1cb conflict fixed
|\
| * 75a857c AND simple
* | 400b400 & simple
|/
* fec145a branch test
```

最后，放心的删除 linuxprobe 分支吧：

```
[root@linuxprobe linuxprobe]# git branch -d linuxprobe
Deleted branch feature1 (was 75a857c).
[root@linuxprobe linuxprobe]# git branch
```

### 21.4 部署 Git 服务器

Git 是分布式的版本控制系统，我们只要有了一个原始 Git 版本仓库，就可以让其他主机克隆走这个原始版本仓库，从而使得一个 Git 版本仓库可以被同时分布到不同的主机之上，并且每台主机的版本库都是一样的，没有主次之分，极大的保证了数据安全性，并使得用户能够自主选择向那个 Git 服务器推送文件了，其实部署一个 git 服务器是非常简单的事情，我们需要用到两台主机，分别是：

编辑

| 主机名称 | 操作系统 | IP 地址 |
| --- | --- | --- |
| Git 服务器 | 红帽 RHEL7 操作系统 | 192.168.10.10 |
| Git 客户端 | 红帽 RHEL7 操作系统 | 192.168.10.20 |

首先我们分别在 Git 服务器和客户机中安装 Git 服务程序(刚刚实验安装过就不用安装了)：

```
[root@linuxprobe ~]# yum install git
Loaded plugins: langpacks, product-id, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.
```

```
Package git-1.8.3.1-4.el7.x86_64 already installed and latest version
Nothing to do
```

然后创建 Git 版本仓库，一般规范的方式要以.git 为后缀：

```
[root@linuxprobe ~]# mkdir linuxprobe.git
```

修改 Git 版本仓库的所有者与所有组：

```
[root@linuxprobe ~]# chown -Rf git:git linuxprobe.git/
```

初始化 Git 版本仓库：

```
[root@linuxprobe ~]# cd linuxprobe.git/
[root@linuxprobe linuxprobe.git]# git --bare init
Initialized empty Git repository in /root/linuxprobe.git/
```

其实此时你的 Git 服务器就已经部署好了，但用户还不能向你推送数据，也不能克隆你的 Git 版本仓库，因为我们要在服务器上开放至少一种支持 Git 的协议，比如 HTTP/HTTPS/SSH 等，现在用的最多的就是 HTTPS 和 SSH，我们切换至 Git 客户机来生成 SSH 密钥：

```
[root@linuxprobe ~]# ssh-keygen
将客户机的公钥传递给 Git 服务器：
[root@linuxprobe ~]# ssh-copy-id 192.168.10.10
root@192.168.10.10's password:
Number of key(s) added: 1
Now try logging into the machine, with:   "ssh '192.168.10.10'"
and check to make sure that only the key(s) you wanted were added.
```

此时就已经可以从 Git 服务器中克隆版本仓库了（此时目录内没有文件是正常的）：

```
[root@linuxprobe ~]# git clone root@192.168.10.10:/root/linuxprobe.git
Cloning into 'linuxprobe'...
warning: You appear to have cloned an empty repository.
[root@linuxprobe ~]# cd linuxprobe
[root@linuxprobe linuxprobe]#
```

初始化下 Git 工作环境：

```
[root@linuxprobe ~]# git config --global user.name "Liu Chuan"
[root@linuxprobe ~]# git config --global user.email "root@linuxprobe.com"
[root@linuxprobe ~]# git config --global core.editor vim
```

向 Git 版本仓库中提交一个新文件：

```
[root@linuxprobe linuxprobe]# echo "I successfully cloned the Git repository" > readme.txt
[root@linuxprobe linuxprobe]# git add readme.txt
[root@linuxprobe linuxprobe]# git status
# On branch master
# Initial commit
# Changes to be committed:
#   (use "git rm --cached ..." to unstage)
#
#         new file:   readme.txt
#
[root@linuxprobe linuxprobe]# git commit -m "Clone the Git repository"
[master (root-commit) c3961c9] Clone the Git repository
 Committer: root
1 file changed, 1 insertion(+)
 create mode 100644 readme.txt
[root@linuxprobe linuxprobe]# git status
# On branch master
nothing to commit, working directory clean
```

但是这次的操作还是只将文件提交到了本地的 Git 版本仓库，并没有推送到远程 Git 服务器，所以我们来定义下远程的 Git 服务器吧：

```
[root@linuxprobe linuxprobe]# git remote add server root@192.168.10.10:/root/linuxprobe.git
```

将文件提交到远程 Git 服务器吧：

```
[root@linuxprobe linuxprobe]# git push -u server master
```

```
Counting objects: 3, done.
Writing objects: 100% (3/3), 261 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To root@192.168.10.10:/root/linuxprobe.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from server.
```

为了验证真的是推送到了远程的 Git 服务，你可以换个目录再克隆一份版本仓库（虽然在工作中毫无意义）：

```
[root@linuxprobe linuxprobe]# cd ../Desktop
[root@linuxprobe Desktop]# git clone root@192.168.10.10:/root/linuxprobe.git
remote: Counting objects: 3, done.
Receiving objects: 100% (3/3), done.
[root@linuxprobe Desktop]# cd linuxprobe/
[root@linuxprobe linuxprobe]# cat readme.txt
I successfully cloned the Git repository
```

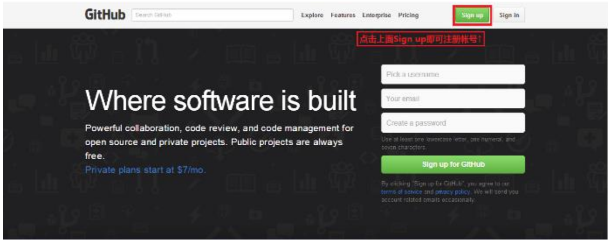
## 21.5 Github 托管服务

其实自己部署一台 Git 服务器或许很有意思，但想到你要保证这台主机能够 7*24 小时稳定运行，还要保证各种权限及版本库的安全就觉得很累吧，

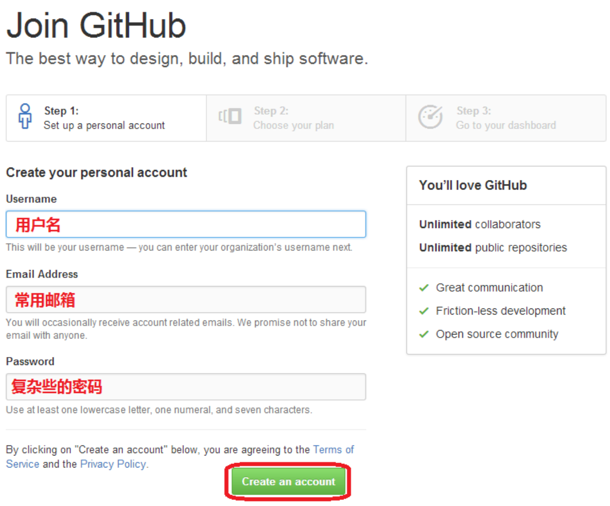



Github 顾名思义是一个 Git 版本库的托管服务，是目前全球最大的软件仓库，拥有上百万的开发者用户，也是软件开发和寻找资源的最佳途径，Github 不仅可以托管各种 Git 版本仓库，还拥有了更美观的 Web 界面，您的代码文件可以被任何人克隆，使得开发者为开源项贡献代码变得更加容易，当然也可以付费购买私有库，这样高性价比的私有库真的是帮助到了很多团队和企业。

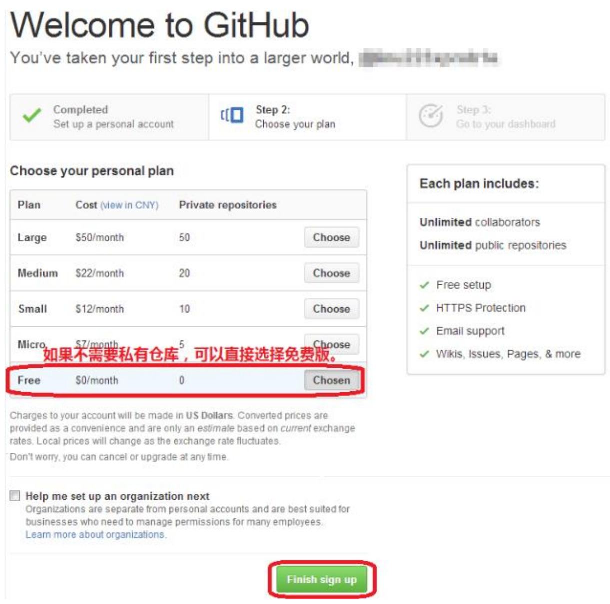
**大多数用户都是为了寻找资源而爱上 Github 的，首先进入网站，点击注册(Sign up)：**



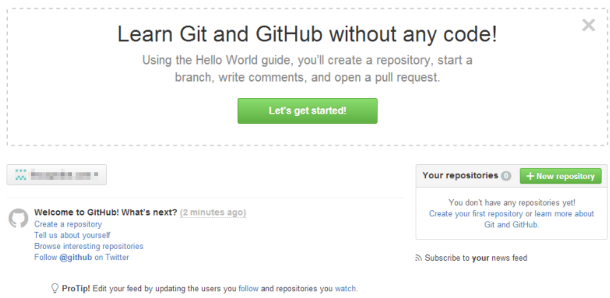
填写注册信息：




选择仓库类型：



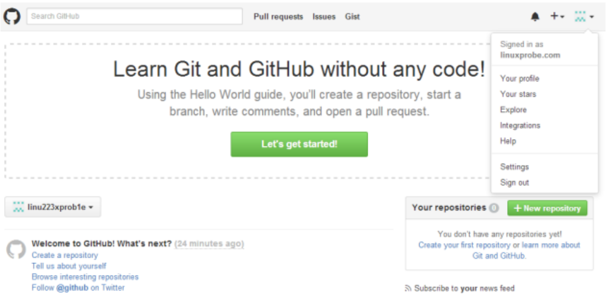

好棒，我们的 GitHub 帐号注册完成了：



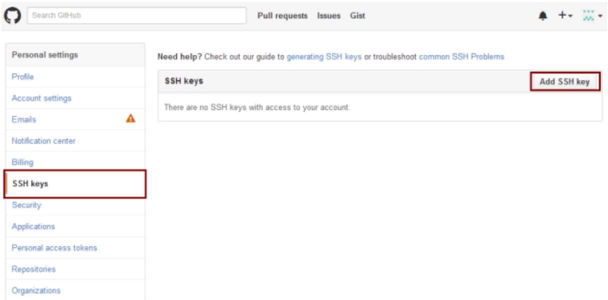

我们在向 Github 推送文件时，可以选择 SSH 协议模式，在本机生成密钥文件（上面实验已经做过，就不需要再生成了）：

```
[root@linuxprobe ~]# ssh-keygen
[root@linuxprobe ~]# ssh-add
```

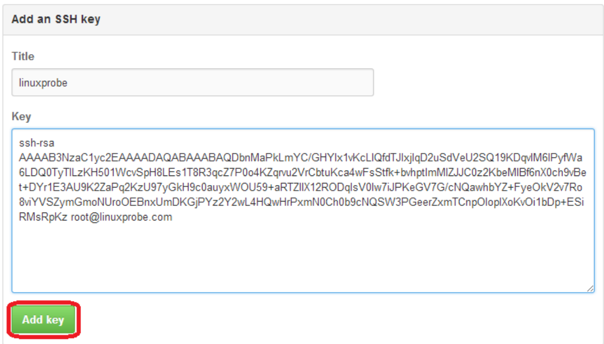
点击进入 Github 的帐户配置页面：



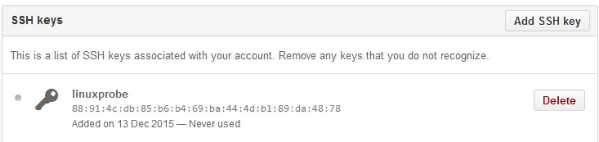

点击添加 SSH 公钥：



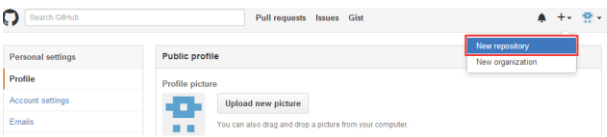

将本机中的 ssh 公钥(.ssh/id_rsa.pub)复制到页面中，填写 ssh 公钥信息：



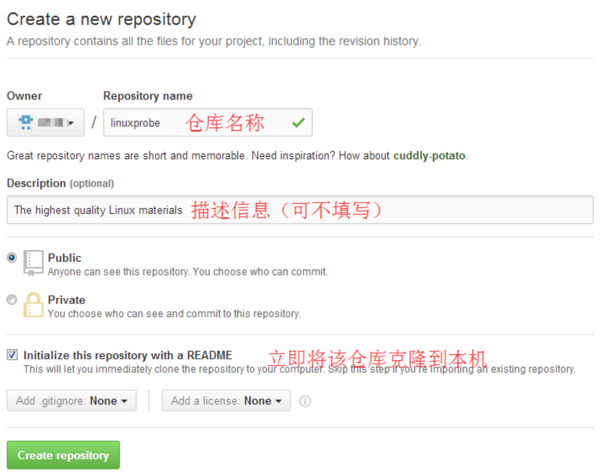

查看 ssh 公钥信息：



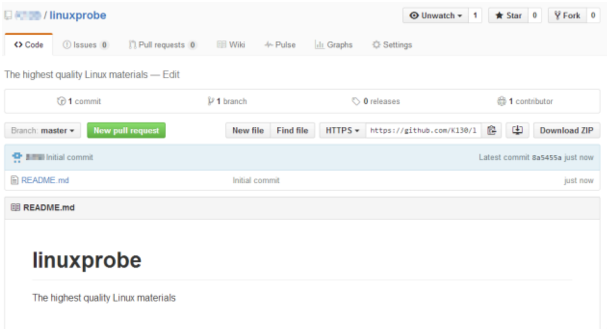

好的，现在我们的准备工作已经妥当，让我们在 Github 上创建自己第一个 Git 版本仓库吧，点击创建一个新的版本仓库：



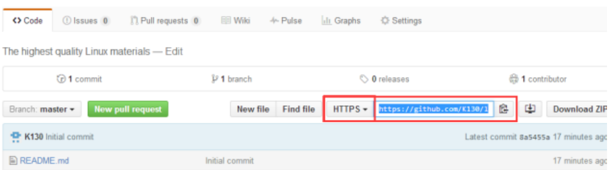

填写版本仓库的信息：




**创建成功后会自动跳转到该仓库的页面：**




复制版本仓库的克隆地址：



尝试把版本仓库克隆到本地(这个版本库我会一直保留，大家可以动手克隆下试试。)：

```
[root@linuxprobe ~]# git clone  https://github.com/K130/linuxprobe.gitCloning into 'linuxprobe'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
[root@linuxprobe ~]# cat linuxprobe/
.git/      README.md
[root@linuxprobe ~]# cat linuxprobe/README.md
# linuxprobe
```

```
The highest quality Linux materials
```

将该 Github 版本仓库添加到本机的远程列表中：

```
[root@linuxprobe linuxprobe]# git remote add linuxprobe git@github.com:K130/linuxprobe.git
[root@linuxprobe linuxprobe]# git remote
linuxprobe
origin
```

编写一个新文件：

```
[root@linuxprobe ~]# cd linuxprobe/
[root@linuxprobe linuxprobe]# echo "Based on the RHEL&Centos system" > features
```

将该文件提交到本地的 Git 版本仓库：

```
[root@linuxprobe linuxprobe]# git add features
[root@linuxprobe linuxprobe]# git commit -m "add features"
```

然后将本地的 Git 仓库同步到远程 Git 服务器上(第一次请加上参数-u，代表关联本地与远程)：

```
[root@linuxprobe linuxprobe]# git push -u linuxprobe  masterCounting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 303 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@github.com:K130/linuxprobe.git
   8a5455a..f1bc411  master -> master
Branch master set up to track remote branch master from linuxprobe.
```

刷新一下 Web 页面，果然看到版本仓库已经同步了：

